HACK X CRACK: NMAP - EL MEJOR ESCANEADOR DE INTERNET



PASO a PASO

Serie RAW: IRC

INTERNET RELAY CHAT



CHATEANDO MEDIANTE TELNET

connecting



Permisos de archivo
CAT y otros comandos

APACHE



Modulos y Servidores Virtuales

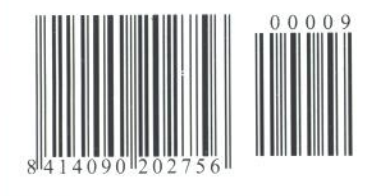
DESCUBRE LOS PROTOCOLOS DE "LA RED" CON NOSOTROS



VISUAL BASIC

Creando Librerias
Acceso a datos

Nº 9 -- P.V.P. 4,5 EUROS



HACK X CRACK - HACK X CRACK - HACK X CRACK - HACK X CRACK

LOS CUADERNOS DE HACK X CRACK
www.hackxcrack.com

NMAP

Instalación
Interfaz Gráfica
Opciones de Uso

HEMOS PUESTO UN SERVIDOR A TU DISPOSICIÓN
HACKEANOS !!!

Iniciación al PORT SCANNING
Técnicas de "scaneo"

ESCANEANDO LA RED

Si no entiendes ni una palabra de esta portada, has llegado a LA REVISTA DE INFORMÁTICA QUE ESTABAS ESPERANDO!!!

PC PASO A PASO: APRENDE A PROGRAMAR CON NOSOTROS

EDITORIAL:

GRACIAS A TODOS

Si eres lector habitual o miembro de EL FORO de Hack x Crack (www.hackxcrack.com) ya sabes que debido a causas mayores AZIMUT me ha sustituido como director durante los números 7 y 8.

No tengo palabras para agradecerle el gran trabajo que ha realizado sacando adelante esta revista. Hace pocos días AZIMUT me pidió que cuando escribiese esta editorial no le mencionase a él sino (en sus palabras) "a los verdaderos artífices del milagro: los colaboradores". Es cierto, debemos agradecerle que sigamos en la "palestra" a muchas personas que han colaborado con HXC de forma totalmente desinteresada, a los miembros del foro que nos ayudan con sus críticas, a los moderadores que mantienen un lugar donde hablar libremente, a los maquetadores que han trabajado noches enteras y con prisas, y no hace falta seguir porque la lista es muy larga y podría dejarme a alguien.

GRACIAS, GRACIAS y GRACIAS.

Espero que este número 9 os guste tanto como los dos anteriores y os anuncio que se preparan cambios importantes que ayudarán a que esta revista llegue tan lejos como todos deseamos: Ya sabes que en EL FORO anunciamos las novedades, pásate de vez en cuando y lee la sección de COMUNICADOS.

Un enorme saludo a todos y os dejo con la revista.

INDICE

GNU LINUX (II)
EL SISTEMA DE ARCHIVOS

1.- Introducción

Habiendo presentado en artículos anteriores el sistema GNU/LiNUX, vamos a ir profundizando poco a poco en los aspectos que lo definen. En concreto en el presente artículo se tratará el Sistema de Archivos de este Sistema Operativo (S.O.). En el próximo artículo pasaremos a describir diversas tareas administrativas como administración de usuarios, dispositivos, etc...

Aunque parece fuera de toda lógica, la razón de posponer estos temas a un tercer capítulo es la siguiente: Tanto la administración de usuarios, como la de servicios o la de dispositivos requiere de un conocimiento al menos básico del Sistema de Archivos dado que en este recaerán cosas tan importantes como permisos, cuotas, etc....

Centrándonos ya en el tema que hoy tratamos, no cabe duda de que tratar con archivos y directorios así como comprender la estructura en la que estos se distribuyen en el disco es algo de importancia vital. No ya para comprender GNU/LiNUX sino cualquier S.O.

Tal como dijimos en pasados artículos, GNU/LiNUX es un S.O. deudor de UNiX y como veremos a lo largo de este artículo, esto también se notará en su Sistema de Archivos. Una vez más, el objetivo es que lo que aprendamos en el siguiente artículo no se limite a la *distro XXXX*... Es más, si después de una lectura atenta del artículo hemos aprendido algo, esto que hayamos aprendido será aplicable a cualquier sistema UNiX.

2.- Archivos

El concepto básico de un archivo, con el que casi todos estamos probablemente familiarizados, define a "archivo" como un conjunto de datos independientes que reside en nuestro disco duro. Con "independientes" queremos decir que podemos tener diversos archivos cuyos contenidos son distintos. Esto hará que sea necesaria la identificación de cada uno de estos archivos de manera que podamos identificar el contenido del mismo frente a los demás.

En GNU/LiNUX identificaremos un archivo por su nombre y su ubicación. En cada ubicación o "directorio" sólo podrá existir un único archivo con un determinado nombre. Esto quiere decir que si hemos guardado nuestros datos en un archivo llamado *ejemplo.txt* y luego queremos guardar más datos en la misma ubicación, tendremos que guardarlos en un archivo con un nombre distinto como *ejemplo2.txt*.

2.1.- Tipos de archivo

Los archivos pueden contener diversos tipos de información. Básicamente podremos clasificar los archivos conforme a su contenido de la siguiente manera:

1.- Datos de usuario: Información que crea y actualiza un usuario. Puede ir desde un simple archivo de texto, hasta archivos más complicados generados por aplicaciones de usuario tales como aplicaciones CAD, procesadores de texto, etc....

2.- Datos del sistema: Información, normalmente en formato de texto plano, que contienen información sobre como está configurado nuestro sistema. Un ejemplo de este tipo de archivos será el archivo */etc/passwd* que contiene información relevante sobre las cuentas de usuario. Obviamente este tipo de archivos es manejado habitualmente por el administrador del sistema, estando vedado su uso al resto de los usuarios.

3.- Archivos ejecutables: Estos archivos contienen instrucciones que el ordenador puede comprender e interpretar. Son los llamados programas o ejecutables.

4.- Archivos de dispositivo: Como se dijo en el artículo anterior, una característica que determina el comportamiento de los sistemas UNiX-like, es que incluso el acceso al ***hardware*** de nuestro ordenador se puede realizar mediante acceso a archivos. Estos archivos un tanto particulares requerirán de un análisis en mayor profundidad que será abordado en breve.

2.2.- Nombres de archivo

En GNU/LiNUX podemos utilizar nombres de archivo de hasta 256 caracteres. Estos caracteres pueden ser tanto letras mayúsculas como letras minúsculas,

números y otros caracteres especiales como el guión (-), el subrayado (_) o el punto (.).
En este punto conviene llamar la atención sobre un aspecto particular de los sistemas UNiX-like: Los nombres de archivos/directorios son ***case-sensitive***; es decir, nombres de archivo que son iguales en apariencia pero difieren en las mayúsculas y las minúsculas serán realmente dos nombres de archivo distintos. Así por ejemplo los siguientes nombres de archivos pueden coexistir en una misma ubicación dado que son archivos distintos: ***ejemplo.txt, ejemplo.TXT, Ejemplo.txt, EJEMPLO.TXT, EjEmPlO.txt, 3j3mp10_h4ch0r_d3_3s0s.txt....***
Quitando el último ejemplo que es una dedicación personal a las reinonas del glam que pululan por doquier, creo que queda bien reflejado el sentido del párrafo anterior.
Existirán así mismo una serie de meta caracteres que no podrán formar parte del nombre de un archivo. Es el caso de los asteriscos (*), signo de interrogación (?), barra invertida (\), etc... Debido a que tendrán un significado concreto para la ***shell***. Cuando se aborde en profundidad la ***shell*** y sus intríngulis veremos más de estos meta caracteres.

3.- Directorios

GNU/LiNUX, al igual que muchos SS.OO. organizan los archivos en directorios. Podemos pensar en los directorios como en carpetas que contienen archivos o a su vez más carpetas.

3.1.- Directorios Padre y Subdirectorios

Imaginemos la siguiente situación: Un directorio A que contiene un directorio B. Llamaremos a B subdirectorio del directorio A, y el directorio A será el directorio padre del directorio B.
Esto que a algunos les parecerá una perogrullada (sobre todo a los que, como servidor, peinan canas) he decidido ponerlo, no ya para rellenar espacio como barruntan las mentes malpensantes, sino como una necesidad perentoria. ¿Lo qué? Pues eso; como tengo la fortuna de dedicarme a esto de la enseñanza de ***rerum calculum machinae***, puedo asegurar que de los nacidos en el 85 en adelante casi nadie ha utilizado en su vida algo similar a una línea de comandos y que incluso pasar del concepto de "carpeta" al de directorio a menudo trae problemas.

3.2.- El directorio raíz: La madre (¿padre?) de todos los directorios

En GNU/LiNUX el directorio que contiene todos los demás directorios es llamado "directorio raíz". Cualquier otro directorio será subdirectorio de este. A partir de este directorio raíz, tras poner un subdirectorio tras otro, formamos una estructura arborescente denominada "árbol de directorios" (ver fig 1 al final de este artículo)

3.3.- Nombres de directorio

Los directorios se nombran de la misma manera que los archivos; es decir siguiendo la misma regla de caracteres que se pueden utilizar y los que no y observando el ***case-sensitive***.
El carácter barra inclinada (/) es utilizado para referirnos a directorios o archivos que están dentro de un directorio. Por ejemplo, ***artículoshxc/artículo2.abw***, nos dice que el archivo ***artículo2.abw*** se encuentra dentro del directorio ***artículoshxc***. Así mismo con el siguiente ejemplo ***artículoshxc/grafs*** nos estamos refiriendo a un subdirectorio, grafs que se encuentra bajo el directorio ***artículoshxc***.
Nótese que a priori es difícil distinguir cuando nos estamos refiriendo a un archivo o a un directorio. Más adelante veremos que mediante los comandos adecuados y parámetros no menos adecuados podremos distinguir fácilmente cuando listamos archivos o directorios. Por ahora puede ser un ejercicio interesante observar las distintas salidas que provocan las siguientes invocaciones de comando:

00 $ ls
01 $ ls -F
02 $ ls --color
03 $ ls --color -F

En cuanto a la salida es probable que muchos no observéis diferencias. Esto es debido a que sobre todo las dos últimas opciones (02 y 03) vienen ya activadas "de serie". Por supuesto nosotros aprenderemos a "activar opciones de serie" cuando nos enfrentemos al artículo "***La shell y tú: una historia de lujuria***".
El directorio raíz será siempre mostrado con un carácter / en vez de referirnos a él con un nombre. No debe causarnos confusión el distinguir cuando el carácter / se utiliza para separar nombres de directorios o cuando se utiliza para referirse al directorio raíz. En este último caso jamás irá precedido de un nombre. Así al escribir ***/usr/bin*** nos estamos refiriendo al directorio ***bin*** que cuelga del directorio ***usr*** que a su vez cuelga del directorio / (raíz).

3.4- El directorio Home

GNU/LiNUX proporciona a cada usuario su propio directorio. Este es el llamado "directorio ***home***" o "directorio casa". Dentro de este directorio cada usuario puede guardar sus propios archivos y crear sus propios subdirectorios.
Dependiendo del sistema GNU/LiNUX que se utilice, los demás usuarios podrán acceder o no a los archivos de un usuarios determinado.
La localización del directorio ***home*** puede ser cambiada por el administrador, pero habitualmente este tipo de directorios se encuentran bajo el directorio ***/home***.

4.- Navegando por el Sistema de Archivos de GNU/LiNUX

Navegar por el sistema de archivos GNU/LiNUX es una tarea harto sencilla: tan sólo tendremos que aprender dos comandos y uno de ellos ni siquiera posee parámetros.

4.1.- El comando pwd: ¿Dónde estoy?

Tecleemos el comando pwd. Observaremos algo como:

00 luis@el_chaman ~ $ pwd
01 /home/luis
02 luis@el_chaman ~ $

Esto quiere decir que nos encontramos en el directorio ***/home/luis***. Este directorio (en el que nos encontremos en cada momento) será conocido como directorio de trabajo o directorio actual. Este es el directorio ***home*** para el usuario ***luis***. Por defecto, cuando entramos en nuestro sistema, GNU/LiNUX nos deposita en nuestro directorio ***home***.

El término ***pwd*** corresponde al acrónimo ***print working directory***. Esto nos muestra otra costumbre bastante común en los entornos UNiX-like: la abreviación de comandos de manera que sean fáciles de teclear.

Una vez que estamos situados en un directorio, querremos saber que contiene. Para ello, se utiliza el comando ***ls***.

00 luis@el_chaman ~ $ ls
01 artículoshxc cosa multimedia public_html
02 luis@el_chaman ~ $

El comando ***ls*** (list) nos lista el contenido del directorio de trabajo. Para ver las numerosas opciones que soporta este comando podemos teclear ***ls --help | more*** (¿quién no se estudió el capítulo de fontanería....? ;o))

4.2.- Nombres de archivo absolutos y relativos

Cuando especificamos el nombre de un archivo, GNU/LiNUX lo busca en el directorio de trabajo. Si el archivo no se encuentra en el directorio de trabajo se nos mostrará un mensaje de error:

00 luis@el_chaman ~ $ ls cosa
01 cosa
02 luis@el_chaman ~ $ ls cosa2
03 ls: cosa: No existe el archivo o directorio

En algunas ocasiones nos interesará referirnos a un archivo que no se encuentre en el directorio de trabajo. Entonces nos tendremos que referir al archivo diciendo el directorio donde se encuentra y el propio nombre del archivo.

00 luis@el_chaman ~ $ ls artículoshxc/artículo2.abw
01 artículo2.abw
02 luis@el_chaman ~ $

En el ejemplo vemos como si queremos listar el archivo ***artículo2.abw*** que se encuentra dentro del subdirectorio ***artículoshxc*** debemos de especificar el nombre del subdirectorio y el nombre del archivo. A esta manera de hacer las cosas se le denomina nombre de archivo relativo dado que la ruta (***path***) hacia el archivo que buscamos es relativa a nuestro directorio de trabajo; es decir, a donde nos encontramos en un determinado momento. Si cambiásemos de directorio de trabajo, este manera de referirnos al archivo ***artículo2.abw*** ya no nos serviría.

Además, puede parecer que esto nos permite realizar llamadas relativas sobre subdirectorios (es decir, directorios que se encuentran por debajo de donde nos encontramos nosotros). Esto no es del todo cierto dado que existen dos directorios especiales que nos permitirán hacer cosas curiosas: . (directorio de trabajo) y .. (directorio padre del directorio de trabajo). Así por ejemplo todas las siguientes llamadas son nombres de archivos relativos al directorio de trabajo actual:

00 luis@el_chaman ~ $ ls ./artículoshxc/artículo2.abw

```
        artículo2.abw
luis@el_chaman ~ $ ls ../tuxed/visio.iso
        visio.iso
```

En el segundo comando .. significa que subimos un nivel, nos metemos dentro del directorio ***home*** del usuario ***tuxed*** y listamos el archivo ***visio.iso***. Por cierto, este archivo nada tiene que ver con cierta aplicación comercial de la que no se puede realizar copias sin el consentimiento expreso de ***MicroSoft***.

La otra manera de referirnos a estos archivos sería mediante el uso de la ruta que va desde el directorio raíz hasta donde se encuentra el archivo que queremos sin tener en cuenta para nada el directorio de trabajo. A esta manera de hacer las cosas se le denomina nombre de archivo absoluto. El ejemplo arriba mostrado quedaría de la siguiente manera:

```
luis@el_chaman ~ $ ls /home/luis/articuloshxc/artículo2.abw
        artículo2.abw
luis@el_chaman ~ $ ls /home/tuxed/visio.iso
        visio.iso
```

4.3.- Nos vamos a hacer turismo por el disco duro: El comando cd

El comando ***cd*** (change directory) nos permitirá cambiar de directorio de trabajo. Es decir, nos moveremos a otro directorio. La sintaxis de este comando es la siguiente:

```
cd <directorio_destino>
```

El espacio entre ***cd*** y ***<directorio_destino>*** es obligatorio. El directorio de destino puede ser el nombre absoluto o relativo de un directorio. Veamos algunos ejemplos:

```
luis@el_chaman $ cd /home/luis      //Camino absoluto
luis@el_chaman $ pwd
        /home/luis
luis@el_chaman $ cd ..              //Camino relativo
luis@el_chaman $ pwd
        /home
luis@el_chaman $ cd luis/articuloshxc//Camino relativo
luis@el_chaman $ pwd
        /home/artículoshxc/
luis@el_chaman $ cd /usr/doc        //Camino absoluto
luis@el_chaman $ pwd
        /usr/doc
luis@el_chaman $ cd /               //Camino absoluto
luis@el_chaman $ pwd
        /
```

El directorio raíz no posee padre, así que si ahora tecleásemos ***cd ..*** (obsérvese el espacio obligatorio entre ***cd*** y los dos puntos) seguiríamos estando en el directorio raíz.

Si tecleamos simplemente ***cd*** sin pasarle argumentos, automáticamente nos lleva a nuestro directorio ***home***:

```
luis@el_chaman $ pwd
        /usr/doc
luis@el_chaman $ cd
luis@el_chaman $ pwd
        /home/luis
```

Es bastante común en las últimas distribuciones que el propio ***prompt*** (el texto que aparece a la derecha) nos diga en todo momento en que directorio nos encontramos. Recordad que no todos los GNU/LiNUX y mucho menos todos los UNiX están configurados de esta manera. Además, siempre es bueno conocer lo que funcionará en todos los sitios. Como anecdotario particular, la salida real en mi sistema de los ejemplos anteriores tiene más bien el siguiente aspecto:

```
luis@el_chaman ~ $ cd /home/luis
luis@el_chaman ~ $ pwd
        /home/luis
luis@el_chaman ~ $ cd ..
luis@el_chaman /home $ pwd
        /home
luis@el_chaman /home $ cd luis/artículoshxc
luis@el_chaman ~/artículoshxc $ pwd
        /home/artículoshxc/
luis@el_chaman ~/artículoshxc $ cd /usr/doc
luis@el_chaman /usr/doc $ pwd
        /usr/doc
luis@el_chaman /usr/doc $ cd /
luis@el_chaman / $ pwd
        /
luis@el_chaman / $ cd
luis@el_chaman ~$ pwd
        /home/luis
```

Obsérvese que por abreviar el directorio home se escribe con el símbolo ~

5.- Creando y borrando archivos

GNU/LiNUX dispone de diversas maneras de crear

y eliminar archivos. De hecho, algunas de estas maneras son tan simples de realizar que muchas veces tenemos que tener cuidado de no borrar o sobrescribir accidentalmente archivos.

Inicialmente vamos a crear un archivo utilizando las redirecciones de entrada/salida. Las redirecciones funcionan de la siguiente manera: Una redirección de salida (>) desvía el flujo de información generado por un programa y que se destina a la salida estándar hacia otra ubicación, normalmente un archivo. Una redirección de entrada (<) desvía el flujo de datos de entrada estándar, es decir el teclado, a otra fuente de datos; normalmente un fichero.

Vamos a crear nuestro primer fichero mandando todo aquello que debería de salir por pantalla a un archivo. Para ello, vamos a teclear la siguiente secuencia de comandos:

```
luis@el_chaman ~ $ cd
luis@el_chaman ~ $ ls -l /bin > ejecutables_en_bin.txt
luis@el_chaman ~ $ ls ejecutables_en_bin.txt
        ejecutables_en_bin.txt
luis@el_chaman ~ $
```

Tras teclear lo arriba mostrado, habremos generado un archivo de texto que contiene un listado de los archivos contenidos en el directorio */bin*. Para que nos hagamos una idea de lo que contendrá ese archivo, basta con que volvamos a ejecutar el comando pero sin redireccionar ahora la salida:

```
luis@el_chaman ~ $ cd
luis@el_chaman ~ $ ls -l /bin
```

Ahora bien; ¿cómo podemos ver el contenido del archivo?. De nuevo existirán varias maneras; listo algunas pero nos quedaremos con la última por ser la más potente:

```
luis@el_chaman ~ $ more ejecutables_en_bin.txt
luis@el_chaman ~ $ less ejecutables_en_bin.txt
luis@el_chaman ~ $ cat ejecutables_en_bin.txt
```

5.1.- cat: Sobre gatos y otros bichos

El comando *cat* es uno de los más simples, útiles y versátiles comandos de los que disponemos en GNU/LiNUX. Para que os hagáis una idea: ¿Cómo creo una imagen ISO de un CDROM? ¿Nero? ¿Algo similar? Agárrense vuesas mercedes los machos y dispongan a asombrarse:

```
luis@el_chaman ~ $ cat /dev/cdrom > imagenCD.iso
```

Pero antes de meternos en tamaños berenjenales, vamos a hablar un poco sobre este gatito. El comando *cat* funciona de la siguiente manera: toma como entrada el primer parámetro que le pasemos y nos muestra su contenido.
Por defecto, si se invoca sin ningún parámetro, se toma como entrada la entrada estándar (es decir, el teclado). Veamos un ejemplo:

```
luis@el_chaman ~ $ cat
        Hola, esto es una prueba. Ahora doy a ENTER
        Hola, esto es una prueba. Ahora doy a ENTER
        y sigo en la siguiente línea.
        y sigo en la siguiente línea.
        ¿Cómo salgo de aquí por Dios?
        ¿Cómo salgo de aquí por Dios?
        <Ctrl-D>
        EOF
```

Parece que nos hemos atascado, compañeros *padawanes*. Cada vez que le damos a ENTER el gatito este se empeña en repetir lo que hemos escrito. Muy bien. Para salir de esta pesadilla lo que tenemos que hacer es presionar la combinación ***Ctrl-D*** (***Control + D***) si estamos al principio de una línea o ***Ctrl-D***, ***Ctrl-D*** (dos veces) si estamos al final. Esto insertará el carácter fin de archivo *(**EOF**,* end of file) con lo que habremos terminado.

Bueno, hemos visto que todo lo que escribíamos se mandaba a la salida estándar. ¿Qué ocurriría si redireccionamos las salida estándar a un fichero?:

```
luis@el_chaman ~ $ cat > nuevo_fichero.txt
        Hola, esto es una prueba. Ahora doy a ENTER
        y sigo en la siguiente línea.
        ¿Cómo salgo de aquí por Dios?
<Ctrl-D>
```

Bueno, parece que el gato este ahora no es tan respondón. Pulsamos de nuevo ***Ctrl-D*** y veremos que volvemos al prompt del sistema.

```
luis@el_chaman ~ $
luis@el_chaman ~ $ cat nuevo_fichero.txt
        Hola, esto es una prueba. Ahora doy a ENTER
        y sigo en la siguiente línea.
        ¿Cómo salgo de aquí por Dios?
luis@el_chaman ~ $
```

Bueno, lo que hemos hecho es utilizar ahora *cat*

como visualizador. Simplemente hemos tecleado *cat nuevo_fichero.txt* y el gatito nos muestra el contenido de dicho fichero. Si quisiéramos añadir más texto teclearíamos:

```
00 luis@el_chaman ~ $ cat >> nuevo_fichero.txt
01     Este es texto añadido
02     <Ctrl-D>
03 luis@el_chaman ~ $ cat nuevo_fichero.txt
04     Hola, esto es una prueba. Ahora doy a ENTER
05     y sigo en la siguiente línea.
06     ¿Cómo salgo de aquí por Dios?
07     Este es texto añadido.
08 luis@el_chaman ~ $
```

Es decir, un único comando de sistema nos puede servir de editor de supervivencia. Eso sí, su uso requiere de mucho cuidado. Fijémonos en la línea 00. La redirección ha sido sustituida por unos >>. Esto quiere que la redirección se comporta como antes pero que añade texto al texto existente. Si sólo pusiéramos un único > sustituiría el antiguo texto por el nuevo (se recomienda experimentar con esto).

Bueno, y podríamos decir que ya hemos terminado con el gatito. Bueno, alguien se preguntará el porqué de que me haya estado refiriendo constantemente a este comando como el gatito. La razón es que *cat* en inglés significa gato. Vale, la tontería del número nueve. ¿Pero por qué demonios llaman a un comando gato? ¿ Por qué no perro, asno,....? Bueno la razón de ello es que cat NO significa gato. El significado de este comando es de nuevo una abreviatura que corresponde a la palabra inglesa concatenate, concatenar. Y esto es porque este comando se construyó para concatenar (unir) archivos. Veamos a cat en acción en el terreno que mejor domina: la concatenación:

```
00 luis@el_chaman ~ $ cat > otro_fichero1.txt
01     Este es un texto modosito que hacemos para
probar una cosa. Versión uno
02     <Ctrl-D>
03 luis@el_chaman ~ $ cat > otro_fichero2.txt
04     Este es un texto modosito que hacemos para
probar una cosa. Versión dos.
05     <Ctrl-D>
06 luis@el_chaman ~ $ cat otro_fichero1.txt
otro_fichero2.txt > otro_fichero3.txt
07 luis@el_chaman ~ $ cat otro_fichero1.txt
08     Este es un texto modosito que hacemos para
probar una cosa. Versión uno
09     Este es un texto modosito que hacemos para
probar una cosa. Versión dos.
10 luis@el_chaman ~ $
```

Como vemos *cat* se ha encargado en la línea 06 los archivos *otro_fichero1.txt y otro_fichero2.txt* mandando el contenido al fichero *otro_fichero3.txt*.

5.2.- Creando directorios

Para crear un nuevo directorio tendremos que usar el comando *mkdir*. La sintaxis de este comando es *mkdir <nombre>* donde *<nombre>* es el nombre que queremos dar al directorio:

```
00 luis@el_chaman ~ $ ls
01     artículoshxc     cosa     multimedia
  nuevo_fichero.txt  otro_fichero1.txt
02     otro_fichero2.txt     otro_fichero3.txt
public_html
03 luis@el_chaman ~ $ mkdir ejercicios_hxc
04 luis@el_chaman ~ $ ls
05     artículoshxc     cosa     ejercicios_hxc
   multimedia  nuevo_fichero.txt  otro_fichero1.txt
06 otro_fichero2.txt     otro_fichero3.txt public_html
```

5.3.- Moviendo y copiando archivos, mareando la perdiz

A menudo necesitaremos mover o copiar archivos. Cabe destacar que el término mover en informática se refiere a realizar una copia de un elemento borrando el original. Copiar, por el contrario, se refiere a hacer una copia conservando el elemento original.

Los comandos que tenemos para mover y copiar son respectivamente:

```
mv <origen> <destino>
cp <origen> <destino>
```

Vamos a ver algunos ejemplos de uso:

```
01 luis@el_chaman ~ $ ls
02     artículoshxc     cosa     ejercicios_hxc
   multimedia  nuevo_fichero.txt  otro_fichero1.txt
03     otro_fichero2.txt     otro_fichero3.txt
public_html
04 luis@el_chaman ~ $ mv ejercicios_hxc ejercicioshxc
05 luis@el_chaman ~ $ ls
06     atículoshxc     cosa     ejercicioshxc
   multimedia  nuevo_fichero.txt  otro_fichero1.txt
```

```
07 otro_fichero2.txt otro_fichero3.txt
public_html
08 luis@el_chaman ~ $ cp nuevo_fichero.txt
nuevo_fichero.bck
09 luis@el_chaman ~ $ ls
10 artículoshxc cosa ejercicioshxc
  multimedia nuevo_fichero.txt nuevo_fichero.bck
11 otro_fichero1.txt otro_fichero2.txt
otro_fichero3.txt public_html
```

En la línea 04 lo que hacemos es mover el directorio que creamos anteriormente; equivale a renombrarlo. En la línea 08 realizamos una copia de *nuevo_fichero.txt* a *nuevo_fichero.bck.*

El uso común de estos comandos suele ser utilizarlos junto con comodines. Los comodines son una serie de caracteres especiales que sustituyen partes del nombre de un archivo (*) o letras del mismo (?). Así los siguientes comandos:

```
01 luis@el_chaman ~ $ cp * /tmp
02 Omitiendo artículoshxc: Es un directorio
03 Omitiendo cosa: Es un directorio
04 Omitiendo ejercicioshxc: Es un directorio
05 Omitiendo multimedia: Es un directorio
06 Omitiendo public_html: Es un directorio
07 luis@el_chaman ~ $ mv *.txt ejercicioshxc
08 luis@el_chaman ~ $ cp *.bc? ejercicioshxc
```

En la línea 01, copiamos TODOS los archivos (no directorios) que hay en el directorio de trabajo al directorio */tmp*. En la línea 07 movemos todos los archivos que terminen en *.txt* al directorio *ejercicioshxc* y en la línea 08 copiamos todos los archivos que terminen en *.bc?* siendo ? cualquier carácter (por ejemplo *nuevo_fichero.bck* o *mikasa.bcx*).

Podemos así mismo mover directorios a otra ubicación o copiar estructuras enteras de directorios:

```
01 luis@el_chaman ~ $ mv cosa ejercicioshxc
02 luis@el_chaman ~ $ ls
03 artículoshxc ejercicioshxc multimedia
   nuevo_fichero.bck public_html
04 luis@el_chaman ~ $ ls ejercicioshxc
05 cosa nuevo_fichero.bck nuevo_fichero.txt
 otro_fichero1.txt otro_fichero2.txt otro_fichero3.txt
06 luis@el_chaman ~ $ cp multimedia ejercicioshxc -
R
07 luis@el_chaman ~ $ ls
08 artículoshxc ejercicioshxc
   multimedia nuevo_fichero.bck public_html
09 luis@el_chaman ~ $ ls ejercicioshxc
10 cosa multimedia nuevo_fichero.bck
nuevo_fichero.txt otro_fichero1.txt otro_fichero2.txt
otro_fichero3.txt
11 luis@el_chaman ~ $
```

En la línea 01 movemos el directorio *cosa* (y todo lo que contenga) al directorio *ejercicioshxc*, quedando dentro de este. En 06 sin embargo hacemos una copia de ***multimedia*** y todo lo que contenga recursivamente y la situamos dentro de *ejercicioshxc*. Obsérvese el parámetro -R (de recursive, recursivo) que es el que indica que se debe de copiar el directorio y todo sus contenidos (incluyendo posibles subdirectorios). Podemos consultar más parámetros de ***mv*** o ***cp*** tecleando ***man mv*** o ***man cp***

5.4.- Borrando archivos y directorios: Jugamos con fuego.

Antes de nada asegurémonos que no estamos como usuario ***root*** (administrador) dado que si metemos la pata y borramos algo que no debemos la cosa tendrá difícil solución. Sin embargo si un usuario normal intentase borrar archivos delicados para el sistema, éste no le dejaría. En próximos apartados veremos la razón de esto.

El comando por antonomasia para el borrado es ***rm*** (de remove, eliminar) cuya sintaxis es: ***rm <nombre>*** donde ***nombre*** es el nombre de archivo a borrar. Al igual que ***cp*** y ***mv*** admite comodines lo cual puede resultar peligroso. Veamos algunos ejemplos:

```
01 luis@el_chaman ~ $ cd ejercicioshxc
02 luis@el_chaman ~/ejercicioshxc $ ls
03 cosa multimedia nuevo_fichero.txt
nuevo_fichero.txt otro_fichero1.txt otro_fichero2.txt
 otro_fichero3.txt
04 luis@el_chaman ~/ejercicioshxc $ rm
nuevo_fichero.txt
05 luis@el_chaman ~/ejercicioshxc $ ls
06 cosa multimedia nuevo_fichero.bck
otro_fichero1.txt otro_fichero2.txt otro_fichero3.txt
07 luis@el_chaman ~/ejercicioshxc $ rm otro_*
08 luis@el_chaman ~/ejercicioshxc $ ls
09 cosa multimedia nuevo_fichero.bck
10 luis@el_chaman ~/ejercicioshxc $
```

Una manera segura de utilizar siempre el comando ***rm*** es añadirle el parámetro -i. Esto hará que cada vez que queramos borrar un determinado archivo

nos pregunte si realmente queremos hacerlo o no:

```
01 luis@el_chaman ~/ejercicioshxc $ ls
02 cosa multimedia nuevo_fichero.bck
03 luis@el_chaman ~/ejercicioshxc $ rm -i nuevo_fichero.bck
04 rm: ¿ borrar "nuevo_fichero.bck"? (s/n) n
05 luis@el_chaman ~/ejercicioshxc $
```

Esta opción viene en muchos sistemas activada por defecto para el usuario *root*.
Para eliminar directorios emplearemos el comando *rmdir* cuya sintaxis es *rmdir <nombre_directorio>*. Antes de poder eliminar un directorio nos debemos de asegurar de que este esté vacío:

```
01 luis@el_chaman ~/ejercicioshxc $ rmdir cosa
02 rmdir: cosa: El directorio no está vacío
03 luis@el_chaman ~/ejercicioshxc $ rm cosa/*
04 luis@el_chaman ~/ejercicioshxc $ rmdir cosa
```

En ocasiones esta manera de borrar un directorio puede resultar extremadamente tediosa, sobre todo si el directorio a borrar contiene múltiples niveles de subdirectorios. Para poderlo hacer con un único comando emplearemos las opciones -r (recursive) y -f (force) del comando *rm*:

```
01 luis@el_chaman ~/ejercicioshxc $ rm -rf cosa
```

Mucho cuidado con este comando. Imaginaos que ocurre si como root ejecutamos

rm -rf /. **No ejecutar**

6.- Permisos y propietarios de los ficheros

Todos y cada uno de los archivos y directorios de un sistema GNU/LiNUX poseen permisos. Podremos cambiar los permisos y en ocasiones los propietarios de nuestros archivos con el fin de proporcionar mayor o menor grado de acceso a los mismos. Los permisos también determinan si los archivos pueden ejecutarse como un comando o no.

Si tecleamos ls -l, veremos una salida similar a esta:

```
01 luis@el_chaman ~/ejercicioshxc $ ls -l
02 drw-r--r-- 1 luis usr 4,0K 2003-04-15 23:00 multimedia
```

La cadena *drw-r--r--* representa los permisos del directorio (podría ser perfectamente un archivo normal) multimedia. El primer *luis* significa el propietario del archivo/directorio y el segundo *usr* representa al grupo de usuarios al que pertenece el archivo/directorio.

6.1.- Cambiando propietarios.

Imaginemos la siguiente situación: deseamos que un archivo del ordenador pase a pertenecer a otra persona:

```
01 luis@el_chaman ~/ejercicioshxc $ cd /usr/share/pixmaps
02 luis@el_chaman /usr/share/pixmaps $ ls -l
03 -rw-r--r-- 1 pepe raros 4,0K 2003-04-15 23:00 logo.png
```

En el directorio */usr/share/pixmaps* tenemos un archivo llamado *logo.png* que pertenece al usuario *pepe* y al grupo *raros*. Queremos cambiar el propietario a *luis* y el grupo a *users*. Lo tendremos que hacer como *root*:

```
01 luis@el_chaman ~/ejercicioshxc $ su --
02 Password:
03 root@el_chaman/usr/share/pixmaps $ chown luis.users logo.png
04 root@el_chaman /usr/share/pixmaps $ ls -l
05 -rw-r--r-- 1 luis users 4,0K 2003-04-15 23:00 logo.png
```

En el ejemplo se cambia a la vez el grupo y el directorio. Sería posible cambiar por separado el usuario y el grupo. Ejemplos que hacen esto serían:
chown luis logo.png,
chown .users logo.png.

6.2.- Permisos.

Observemos un momento la siguiente tabla. En ella se nos muestra la notación binaria denominada octal debido a que puede mostrar ocho valores distintos.

Octal	decimal
000	0
001	1
010	2
011	3
100	4
101	5
110	6
111	7

Ahora analicemos un momento la salida que nos dio la ejecución del comando *ls -l* anteriormente ejecutado:

```
05 -rw-r--r-- 1 luis users..........
```

Agrupemos ahora esa cadena de erres, guiones y uves doble en grupos de tres comenzando por la

izquierda. Obtendremos algo parecido a:
rwx
rw- permisos para el propietario al que pertenece el archivo
r-- permisos para el grupo al que pertenece el archivo
r-- permisos para el resto de los usuarios

r: permiso de lectura
w: permiso de escritura
x: permiso de ejecución
-: sin permiso

Teniendo esta información en mente, veamos que sucede cuando cambiamos los permisos de un archivo:

01 root@el_chaman /usr/share/pixmaps $ ls -l
02 -rw-r--r-- 1 luis users 4,0K 2003-04-15 23:00 logo.png
03 root@el_chaman /usr/share/pixmaps $ exit
04 luis@el_chaman /usr/share/pixmaps $ chmod 755 logo.png
05 luis@el_chaman /usr/share/pixmaps $ ls -l
06 -rwxr-xr-x 1 luis users 4,0K 2003-04-15 23:00 logo.png

Vemos como han cambiado los permisos del archivo, ahora bien, ¿cómo lo hemos hecho? ¿Qué significa ese extraño número que he puesto? Fijémonos ahora en la tabla adjunta. Concretamente en los números binarios. Imaginemos que los 1s activan permisos i los 0s los cancelan. Como tenemos que asignar a cada grupo de permisos (propietario, grupo y resto de usuarios) un valor para lectura, escritura y ejecución, lo haremos aplicando a los tres posibles permisos una máscara. Así tendremos:

		máscara		
	Inicial	Dec	Bin	Resultado
Propietario	rwx	7	111	rwx
Grupo	rwx	5	101	r-x
Resto	rwx	5	101	r-x

Observamos que al escribir ***chmod 755*** archivo, lo que hacemos es pasar una máscara binaria a los posibles permisos activándolos o desactivándolos para el archivo. Sólo podremos cambiar los permisos de un archivo si poseemos permiso de escritura sobre él.

Los permisos pueden ser modificados tanto en archivos como en directorios teniendo en cuenta que el concepto de "permiso de ejecución" posee un significado muy distinto para un directorio. Os invito a que investiguéis qué sucede si damos permiso de ejecución o no a un directorio. Utilizad dos usuarios distintos que pertenezcan a distintos grupos. Ya me contaréis vuestras experiencias ;o)

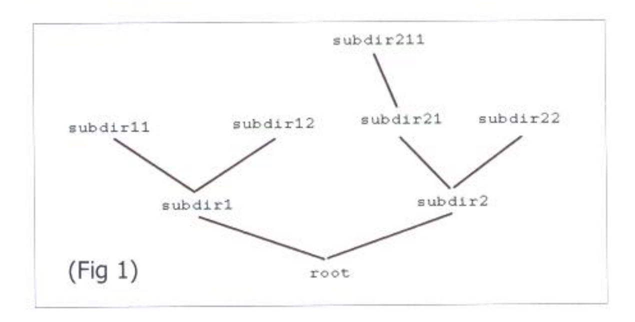
7.- Conclusiones

Una vez más nos quedamos cortos de espacio y muchas cosas en el tintero. Aún así espero haber sintetizado lo más importante de los sistemas de archivos bajo entornos UNiX.

Quede como comentario final que debido al espacio no se ha podido comentar en profundidad la finalidad de cada uno de los directorios. Afortunadamente esta es una información bastante sencilla de encontrar: ***http://www.pathname.com/fhs/2.2/*** En esta página se nos muestra el ***Filesystem Hierarchy Standard*** (Jerarquía Estándar del Sistema de Archivos) o dicho de otra manera: Qué va en cada directorio, cuántos directorios puedo encontrarme en un sistema UNiX , qué puedo esperar encontrarme en cada uno de ellos, etc.....

Una vez leáis esa página comprenderéis el porqué de la omisión de tan interesante apartado (y vital para el tema que tratamos).

Saludos.



(Fig 1)

EL GANADOR DEL SORTEO DE UN SUSE LINUX 8.2 DEL MES DE FEBRERO ES:
OSCAR PONS MARTI
MENORCA
SEGUIR LLAMANDO, EL PROXIMO PODRIA SER PARA TI (PAG 66)

SERVIDOR DE HXC MODO DE EMPLEO

- Hack x Crack ha habilitado un servidor para que puedas realizar las prácticas de hacking.

- Actualmente tiene el BUG del Code / Decode y lo dejaremos así por un tiempo (bastante tiempo ;) Nuestra intención es ir habilitando servidores a medida que os enseñemos distintos tipos de Hack, pero por el momento con un Servidor tendremos que ir tirando (la economía no da para mas).

- En el Servidor corre un Windows 2000 Advanced Server con el IIS de Servidor Web y está en la IP 80.36.230.235.

- El Servidor tiene tres unidades:
* La unidad c: --> Con 2GB
* La unidad d: --> Con 35GB y Raíz del Sistema
* La unidad e: --> CD-ROM

Nota: Raíz del Servidor, significa que el Windows Advanced Server está instalado en esa unidad (la unidad d:) y concretamente en el directorio por defecto \winnt\ Por lo tanto, la raíz del sistema está en d:\winnt\

- El IIS, Internet Information Server, es el Servidor de páginas Web y tiene su raíz en d:\inetpub (el directorio por defecto)

Nota: Para quien nunca ha tenido instalado el IIS, le será extraño tanto el nombre de esta carpeta (d:\inetpub) cómo su contenido. Pero bueno, un día de estos os enseñaremos a instalar vuestro propio Servidor Web y detallaremos su funcionamiento.

De momento, lo único que hay que saber es que cuando TÚ pongas nuestra IP (la IP de nuestro servidor) en tu navegador, lo que estás haciendo realmente es ir al directorio d:\Inetpub\wwwroot\ y leer un archivo llamado default.htm.

Nota: Como curiosidad, te diremos que APACHE es otro Servidor de páginas Web (seguro que has oído hablar de él). Si tuviésemos instalado el apache, cuando pusieses nuestra IP en TU navegador, accederías a un directorio raíz del Apache (donde se hubiese instalado) e intentarías leer una página llamada index.html

Explicamos esto porque la mayoría, seguro que piensa en un Servidor Web como en algo extraño que no saben ni donde está ni como se accede. Bueno, pues ya sabes dónde se encuentran la mayoría de IIS (en \Inetpub\) y cuál es la página por defecto (\Inetpub\wwwroot\default.htm). Y ahora, piensa un poco... ... ¿Cuál es uno de los objetivos de un hacker que quiere decirle al mundo que ha hackeado una Web? Pues está claro, el objetivo es cambiar (o sustituir) el archivo default.html por uno propio donde diga "hola, soy DIOS y he hackeado esta Web" (eso si es un lamer ;)

A partir de ese momento, cualquiera que acceda a ese servidor, verá el default.htm modificado para vergüenza del "site" hacheado. Esto es muy genérico pero os dará una idea de cómo funciona esto de hackear Webs ;)

- Cuando accedas a nuestro servidor mediante el CODE / DECODE BUG, crea un directorio con tu nombre (el que mas te guste, no nos des tu DNI) en la unidad d: a ser posible (que tiene mas espacio libre) y a partir de ahora utiliza ese directorio para hacer tus prácticas. Ya sabes, subirnos programitas y practicar con ellos :)

Puedes crearte tu directorio donde quieras, no es necesario que sea en d:\mellamojuan. Tienes total libertad!!! Una idea es crearlo, por ejemplo, en d:\winnt\system32\default\mellamojuan (ya irás aprendiendo que cuanto mas oculto mejor :)

Es posiblemente la primera vez que tienes la oportunidad de investigar en un servidor como este sin cometer un delito (nosotros te dejamos y por lo tanto nadie te perseguirá). Aprovecha la oportunidad!!! e investiga mientras dure esta iniciativa (que esperamos dure largos años)

- En este momento tenemos mas de 600 carpetas de peña que, como tu, está practicando. Así que haznos caso y crea tu propia carpeta donde trabajar.



MUY IMPORTANTE...

MUY IMPORTANTE!!!!! Por favor, no borres archivos del Servidor si no sabes exactamente lo que estás haciendo ni borres las carpetas de los demás usuarios. Si haces eso, lo único que consigues es que tengamos que reparar el sistema servidor y, mientras tanto, ni tu ni nadie puede disfrutar de él :(Es una tontería intentar "romper" el Servidor, lo hemos puesto para que disfrute todo el mundo sin correr riesgos, para que todo el mundo pueda crearse su carpeta y practicar nuestros ejercicios. En el Servidor no hay ni Warez, ni Programas, ni claves, ni nada de nada que "robar", es un servidor limpio para TI, por lo tanto cuídalo un poquito y montaremos muchos más :)

APACHE PARTE III: CONFIGURACION - COMPARTE TUS FICHEROS MEDIANTE WEB

Bienvenidos de nuevo. En el número anterior se explicó los comandos básicos de Apache. se comentó con ejemplos las principales directivas que hacen funcionar el servidor Apache y se aplicó todo lo aprendido para poder compartir ficheros mediante HTTP sin necesidad de programas externos.
Este capítulo está dividido en dos partes. en la primera parte profundizaremos en algunos módulos y en la segunda parte comenzaremos a crear servidores virtuales. ¿tienes curiosidad de saber qué es todo esto?. ... ya verás.

1. Los módulos de Apache.

Un módulo de Apache es un componente que añade funcionalidad al servidor. Existen diferentes módulos y pueden ser configurables. Es necesario que sepas que puedes crear un módulo programado en C y adaptar el servidor Apache a tus necesidades. En la red encontrarás infinidad de módulos realizados por programadores independientes.

Los módulos se pueden clasificar en 3 categorías:

1. Módulos base: Son los módulos con las funcionas básicas de Apache.

2. Módulos multiproceso: son los responsables de la unión con los puertos del ordenador.

3. Módulos adicionales: cualquier otro módulo que añade funcionalidad pero no es estrictamente necesario para el funcionamiento del Apache.

¿Dónde se encuentran los módulos adicionales?, los localizarás en c:\apache\modules\

Apache instala por defecto los siguientes módulos, es necesario que sepáis por lo menos qué hacen para luego entender la práctica que vamos a realizar.

mod_auth_anon: permite a usuarios anónimos acceder a áreas autentificadas.

mod_auth_dbm: proporciona autentificación utilizando ficheros DBM.

mod_auth_digest: autentificación de usuario utilizando MD5.
mod_cern_meta: Semántica de etiquetas meta del CERN.

mod_expires: Generación de las cabeceras http Expires, de acuerdo de los criterios especificados por el usuario.

mod_headers: personalización de las peticiones HTTP y las cabeceras de las respuestas.

mod_info: proporciona una visión comprensiva de la configuración del servidor.

mod_mime_magic: determina el tipo MIME de un fichero mirando unos pocos bytes del contenido.

mod_proxy: servidor HTTP/1.1 proxy/gateway.

mod_rewrite: proporciona un motor de reescritura basado en reglas que rescribe las peticiones de URL's al vuelo.

mod_speling: intenta corregir las URL mal puestas por los usuarios, ignorando las mayúsculas y permitiendo hasta una falta.

mod_status: proporciona información en la actividad y rendimiento del servidor.

mod_unique_id: proporciona variables de entorno y un identificador único para cada petición.

mod_usertrack: registro de actividad de un usuario en el sitio.

mod_vhost_alias: Proporcionado para configurar muchos servidores virtuales dinámicamente.



DMB, MD5...

DMB, MD5, CERN, Cabeceras HTTP Expires, MIME ... Si eres profano en la materia, seguro que todo eso te suena a "Klingoniano". No te preocupes, ya sabes que cuando necesitamos conocer un término/concepto para la comprensión de un artículo siempre lo explicamos a lo largo del mismo.

Algunos de estos términos abarcan temas bastante extensos y no es necesario explicarlo en este momento puesto que NO IMPIDEN la comprensión de este artículo, pero te recomendamos que cuando tengas un par de horas libres "curiosees" en el mejor buscador que existe actualmente www.google.com y hagas una "toma de contacto" con ellos. Por cierto, si quieres aprender klingon puedes acudir a The Klingon Language Institute (http://www.kli.org/) ;p

Una de las principales razones de emplear módulos en Apache, es que no toda instalación requiere de las mismas funcionalidades, no todas las instalaciones requieren de "Virtual Hosting".

Por lo tanto si fueran incluidas todas las funcionalidades posibles en una versión única de Apache, lo haría sumamente pesado en cuanto a requerimientos de Memoria RAM y espacio en Disco Duro, por esto se opta por "modularizar" e incluir solo lo necesario.

De momento, simplemente vamos a quedarnos con la idea de que estos módulos añaden funcionalidad al Servidor Web y que algunos de ellos son instalados por defecto.

Ahora queremos averiguar cuales son los módulos que están funcionando por defecto en nuestro Servidor Web. Abrimos una consola MSDOS y ponemos el comando de sistema **APACHE –l**, y aparecerá un listado de los módulos que están funcionando. Lo que hace el parámetro **–l** es abrir el archivo de configuración **httpd.conf** y busca los módulos que debe cargar Apache al iniciarse.

```
MS-DOS
10 x 18
C:\apache\Apache>\

C:\apache\Apache>apache -l
Compiled-in modules:
  http_core.c
  mod_so.c
  mod_mime.c
  mod_access.c
  mod_auth.c
  mod_negotiation.c
  mod_include.c
  mod_autoindex.c
  mod_dir.c
  mod_cgi.c
  mod_userdir.c
  mod_alias.c
  mod_env.c
  mod_log_config.c
  mod_asis.c
  mod_imap.c
  mod_actions.c
  mod_setenvif.c
  mod_isapi.c

C:\apache\Apache>
```

2. Listar contenido de los directorios

Apache está considerado como uno de los servidores web más estables que existen e infinitamente más seguro que el servidor IIS de Microsoft. Aún así tiene algunos fallos, ¿no te lo crees?.

Apache necesita utilizar módulos adicionales para mostrar contenido dinámico, y en estos módulos es donde podemos encontrar los fallos, recuerda que cuantos más módulos se instalen más probabilidad hay de tener agujeros de seguridad.

En el número anterior se explico cómo Apache puede listar el contenido del directorio, siempre que nosotros configuremos el archivo httpf.conf para ello. Aún configurando este archivo para que no liste el contenido de los directorios podemos saber los archivos alojados en todos los directorios, incluso en aquellos directorios que tienen una página Index.

Vamos a explicar dos formas de obtener el contenido de los directorios, el primer método será utilizando un gran número de barras inclinadas y el segundo método utilizando un bug del módulo Multiview. Al finalizar cada método daremos respuesta a como evitar que seamos víctimas de estos ataques.

Listado del directorio utilizando barras inclinadas.

Cuando realizamos una petición a un servidor Apache, la URL es procesada por varios módulos, entre ellos mod_negotiation.c, mod_include.c, mod_autoindex.c, estos módulos se encuentran compilados en la versión predeterminada de Apache, es decir, que estos módulos están instalados y activos en la mayoría de los servidores Apache.

Cuando se pasa una URL de gran longitud por estos módulos puede provocar que Apache liste el contenido de sus directorios. Así de simple. Lógicamente los señores de Apache solucionaron el problema cuando fue descubierto, pero aún quedan muchos servidores Apache con este problema. Las versiones hasta 1.3.19 inclusive son vulnerables a este ataque.

¿Es válida cualquier cadena?, no. Tiene que ser una cadena de barras inclinadas y la longitud es variable. Así que no tenemos más remedio que hacer pruebas hasta que Apache liste el directorio. Si consideramos que Apache admite una URL de 8000 caracteres como máximo, podemos listar el contenido en un par de minutos si nos creamos un programa que genere URLs con barras inclinadas.

El código es muy sencillo y supongo que estás aprendiendo Visual Basic gracias al curso de HackxCrack, en los números anteriores del curso de Visual Basic encontrarás todo lo necesario para crear el programa. De todos modos, colocaremos un sencillo programa que hemos realizado para aquellos que no quieran calentarse la cabeza. No estaría mal que cada uno de vosotros hiciera una versión y la publicara en el foro, ¿os atrevéis?

Un ejemplo de cadena válida si queremos saber el contenido del directorio ***http://www.hackxcrack.com/cgi-bin////////////////////////////////////*** (funcionaría en el caso de que el servidor fuese la versión 1.3.19 o anterior)

Pero antes de que os pongáis como locos a poner barras inclinadas es necesario que averigüéis la versión de Apache. En el capítulo anterior se explicaron varias formas de saber remotamente la versión de Apache.

> Mientras escribo este artículo estoy buscando servidores con este bug y he encontrado una empresa de hosting francesa que ofrece espacio web a sus clientes con este bug, ya te puedes imaginar lo que ocurre ¿no?, pues que estoy viendo el listado de todos los archivos del directorio cgi-bin.

Os vamos a explicar otra forma de identificar el servidor web, a modo de culturilla el procedimiento de identificar el servidor se le llama "captura de mensajes" o "banner grabbing". Recordar que no siempre vamos a poder capturar esta información, el servidor web tiene que tener la especificación HTTP 1.1 (RFC 2616), que envía la información que buscamos en los encabezados.



En esta ocasión...

En esta ocasión vamos a utilizar el todopoderoso programa netcat, esta aplicación ya ha sido explicada y utilizada en prácticas anteriores de otros artículos de HxC.

Abre una venta de MSDOS y pon lo siguiente:
nc –nvv 127.0.0.1 80

tras el mensaje que aparece escribe:
HEAD / HTTP/1.0
[Pon dos retornos de carro, es decir, pulsa el botón Return dos veces]

Y verás la información del servidor.

Lo importante es conocer distintas formas de averiguar la información, si estás en un cibercafe y no dispones del netcat pues utiliza el telnet o cualquier herramienta online como dnsstuff.com o netcraft.com, además estas herramientas online tienen una ventaja y es que el servidor consultado no registrará tu IP ;)

```
C:\>nc -nvv 127.0.0.1 80
(UNKNOWN) [127.0.0.1] 80 (?) open
HEAD / HTTP/1.0

HTTP/1.1 200 OK
Date: Fri, 04 Apr 2003 10:56:20 GMT
Server: Apache/1.3.27 (Win32)
Connection: close
Content-Type: text/html

sent 17, rcvd 131: NOTSOCK

C:\>
```



Hemos utilizado la ...

Hemos utilizado la IP 127.0.0.1 y puerto 80 porque se supone que has seguido nuestro curso y en este momento tienes corriendo el apache en tu sistema en el localhost (127.0.0.1) y puerto 80. Si no, ya sabes, al final de la revista tienes el índice de los números anteriores publicados y te indica cómo conseguirlos y, por si no lo sabes, tienes en la Web el Número 1 de Hack x Crack GRATIS en formato PDF… ¿alguien da más? ;p

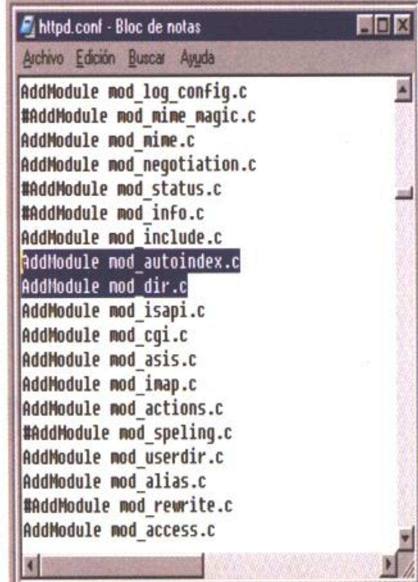
Solución al problema de las barras invertidas

Tenemos dos soluciones: actualizar el servidor web Apache a una versión mayor a 1.3.19 o modificar el fichero de configuración.

Solucionar el problema mediante el archivo de configuración es muy sencillo, abrimos con el bloc de notas el archivo httpd.conf y buscamos las siguientes líneas:

AddModule mod_autoindex.c
AddModule mod_dir.c

Estas líneas son las encargadas de añadir los módulos al iniciarse el Apache, simplemente pon las almohadillas delante y graba el archivo. sabes que poniendo almohadillas delante transformas la línea (instrucción/comando) en un simple texto (comentario), ya lo explicamos en anteriores números.



#AddModule mod_autoindex.c
#AddModule mod_dir.c

Como podemos ver con AddModule configuramos el servidor para que cargue los módulos y así obtener nuevas funcionalidades, en este caso hemos quitado la funcionalidad de mostrar el contenido de los directorios de manera radical. Tienes que tener en cuenta que ya no podremos compartir archivos si hemos anulado estos módulos.



Siempre que ...

Siempre que sea posible se recomienda actualizar la versión del programa afectado por un bug, cualquier otra solución como la explicada es válida pero suele tener consecuencias: una limitación en los accesos, anulación de funciones, etc.

Listado de directorios con Multiview

Existe otro bugs que ya ha sido corregido pero que aún nos lo podemos encontrar en algunos servidor Apache, es un error del parámetro Multiview que permite negociar el contenido basándose en el lenguaje del documento.

Este parámetro no está activo de manera predeterminada así que es más difícil que lo encontremos en el servidor de la victima, pero nunca se sabe, ¿verdad?.

Este parámetro puede estar colocado de la siguiente forma:

Options FollowSymLinks MultiViews

¿Cómo podemos listar el contenido del directorio utilizando el bugs de MultiViews?, es tan sencillo como colocar una URL en el navegador o en el netcat, por ejemplo: *http://127.0.0.1/mp3?M=D*, fíjate que finaliza con *?M=D*.

Esta manera de ver el contenido del directorio no es muy eficaz y es raro de que encuentres un servidor con este error. Pero no está mal saberlo, ¿no crees?

Módulo de configuración "mod_info.c"

El módulo mod_info.c permite saber la configuración del servidor Apache, es información estática pero de gran valor para consultar como está configurado de manera remota. Lógicamente el módulo mod_info.c no está activado de manera predeterminada. Los grandes servidores web, empresas de hosting, ... suelen tener activado este módulo para conocer el estado de la configuración del servidor y ya podemos suponer que a muchos administrados se les olvida proteger o anular el acceso a los usuarios no permitidos

El módulo mod_info da una visión general de la configuración del servidor bastante amplia, incluyendo los módulos y directrices.

Vamos a realizar una práctica de cómo ponerlo en marcha para nuestro uso personal.

Lo primero que hacemos es abrir el fichero de configuración **httpd.conf** y buscamos la cadena **#LoadModule info_module modules/mod_info.so ,** esta cadena dice al Apache que lea el módulo mod_info.so, le quitamos la almohadilla (#) para que realmente sea leído. Ahora buscamos la cadena **#AddModule mod_info.c** y también le quitamos la almohadilla (#), con addmodule le decimos a Apache que añada el módulo mod_info.c



Como has ...

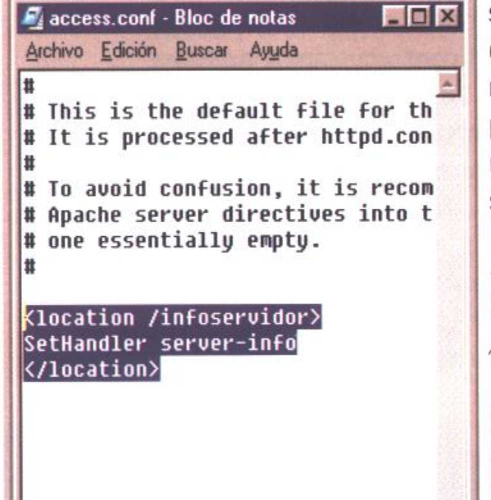
Como has podido comprobar para que Apache tenga en cuenta un módulo primero hay que indicarle que lea el módulo para después sea añadido. Recuerda que son dos pasos "LoadModule y AddModule".

Ya queda poco, ahora tenemos que abrir el archivo access.conf que lo encontrarás en el mismo directorio que el archivo de configuración httpd.conf.

Este archivo también de configuración se utiliza para acceder a los permisos de los archivos, directorios y script del sitio creado. Es un archivo de apoyo, el contenido de este archivo podría estar perfectamente en httpd.conf, pero para no terminar teniendo un mega archivo de configuración se divide la configuración en otros dos archivos (access.conf y srm.conf). De momento nos quedamos con httpd.conf y access.conf, el primero es el archivo principal de configuración y el segundo sirve de apoyo cuyo contenido es definir permisos.

Continuamos con la práctica, abrimos el archivo access.conf (seguramente estará vacío) y ponemos las siguientes líneas:

<location /infoservidor>
SetHandler server-info
</location>



Reiniciamos el servidor Apache (**Apache -k restart**) y ponemos en el navegador la siguiente URL:

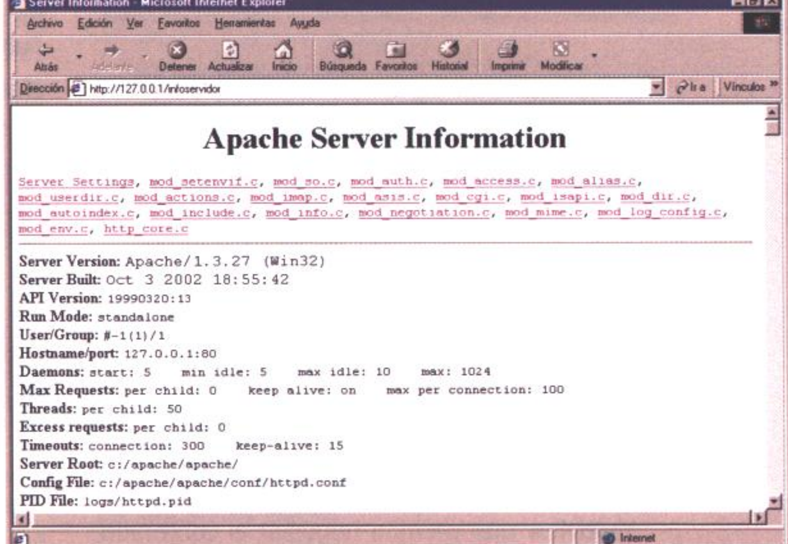
---> *http://127.0.0.1/infoservidor*

El resultado es una página con toda la información de la configuración del servidor Apache.

* Para acceder únicamente a la información del servidor ponemos:
---> *http://127.0.0.1/infoservidor?server*

* Para información de la configuración de los módulos pondremos una dirección tipo **http://127.0.0.1/infoservidor?nombre_del_módulo**, por ejemplo:
---> *http://127.0.0.1/infoservidor?mod_info.c*

* Para obtener la información de los módulos que se están utilizando:
---> *http://127.0.0.1/infoservidor?list*



Por último decir que infoservidor puede ser cambiado por cualquier otra palabra como info, info-servidor, info-server, admin, configure... No estaría mal crear un programa cuyo objetivo sea buscar la url de acceso a la configuración del servidor Apache si es que tiene activado el módulo mod_info.c, ¿alguien se atreve?, con lo aprendido aquí y en el curso de Visual Basic sería suficiente para crear el programa, lo ideal sería utilizar un diccionario, ¿algún valiente?



Pásate por el foro ...

Pásate por EL FORO de la revista (www.hackxcrack.com) y podrás conversar con los lectores y colaboradores de PC Paso a Paso (Los Cuadernos de Hack x Crack). A estas alturas ya formamos una peña de más de 2000 "socios". Te estamos esperando!!! ;)

Módulo de estado del servidor

Apache ofrece otro módulo de gran valor para conocer mediante URL el estado del servidor web. Si es accesible mediante URL quiere decir que podemos acceder a esta información de otros servidores si tienen activado este módulo. Para nosotros será de vital importancia para conocer si el servidor Apache necesita más recursos.

La información que podemos consultar es la siguiente:

- Hora actualizada de sistema del servidor.
- Hora del último reinicio del servidor.
- Tiempo que ha pasado desde que se encendió el servidor.
- Número total de accesos que han tenido lugar desde entonces.
- Bytes totales transmitidos.
- Número total de los thread dedicados a atender peticiones.
- Número total de thread.
- Estado de cada thread, número de peticiones que ha atendido cada uno.
- Promedios con el número de peticiones efectuadas por segundo.
- Porcentaje actual de CPU utilizada por cada thread y el total que utiliza Apache.
- Host y peticiones procesadas.

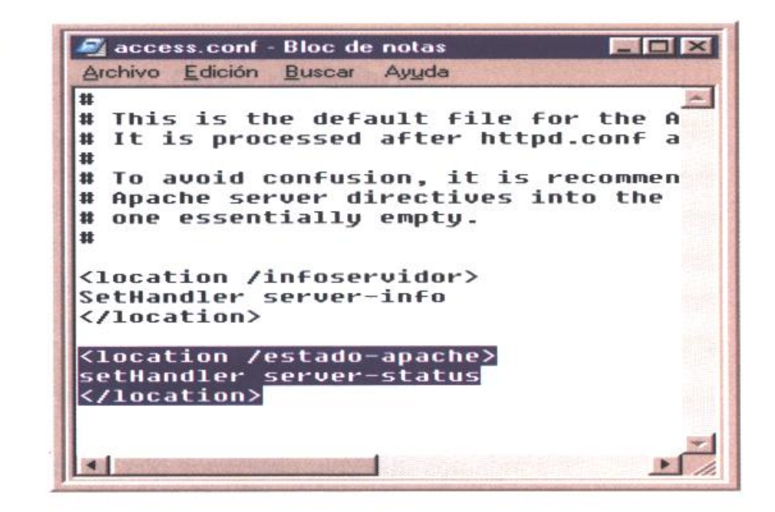
Este módulo no se inicia por defecto, así que habrá que hacer lo mismo que con el módulo mod_info.c. El módulo que tenemos que añadir es **mod_status**, repetimos los pasos anteriores pero esta vez con **mod_status** en el archivo de configuración httpd.conf.

Abrimos el archivo access.conf y añadimos las siguientes líneas:

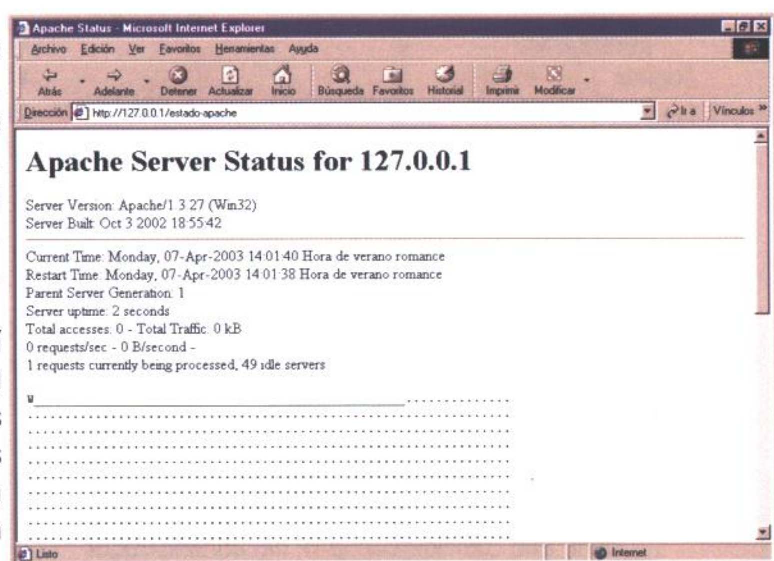
<location /estado-apache>
setHandler server-status
</location>

Recuerda que podemos cambiar estado-apache por otra palabra como estado, infoestado , ... pero para la práctica ponemos estado-apache.

Reiniciamos de nuevo el servidor Apache mediante **Apache –k restart**.



Para acceder al estado del servidor colocamos:
http://127.0.0.1/estado-apache



Puede ocurrir que no aparezcan todos los datos que esperamos, si por ejemplo no aparece la información tan relevante como "Total accesses", "Total Traffic" entre otros datos es necesario volver a modificar el fichero de

configuración httpd.cond, busca la cadena **ExtendedStatus On** y quita la almohadilla.

```
#
# ExtendedStatus controls whether Apache
# information (ExtendedStatus On) or jus
# Off) when the "server-status" handler
#
ExtendedStatus On
```

De momento cualquier navegante que ponga esta URL podrá acceder a la información del estado y configuración del servidor Apache. Desde el punto de vista de la seguridad deja mucho que desear. Apache nos permite configurar el acceso a esta información mediante URL y ponerle limitaciones y para ello hay que configurar de nuevo los comandos que hemos colocado en el archivo access.conf.

Supongamos que solo deseamos que tenga acceso a la información del estado del servidor la IP **254.67.67.5** (nuestra IP al conectarnos a Internet), es tan sencillo como editar el archivo access.conf y colocarlo de la siguiente manera:

<location /estado-apache>
setHandler server-status
order deny,allow
deny from all
allow from 254.67.67.5
</location>

De esta forma tan simple eliminamos el acceso a cualquier navegante a la información del estado del servidor cuando ponga http://tu_ip/estado-servidor excepto para el navegante que tenga la IP 254.67.67.5 (que tiene que coincidir con la IP del administrador del servidor Apache).

Como se ha comentado anteriormente estos módulos son de gran utilidad para el administrador del sistema y suele estar activado para que el administrador en cualquier momento pueda conocer el estado del servidor.



Para ir abriendo ...

Para ir abriendo miras y sabiendo que este acceso suele estar abierto pero protegido por un nombre que no conocemos (en nuestro caso estado-servidor) y por una IP determinada; podríamos "adivinar" la IP determinada que permite el acceso (muchas veces es la misma que la del servidor, aunque otras no ;)) y el nombre por diccionario (hemos propuesto ese ejercicio en este mismo artículo) y utilizando técnicas no explicadas todavía en HXC enviar peticiones de acceso previamente "retocadas".

Algunos avanzados ya saben perfectamente a lo que nos referimos, esas peticiones se "modifican" para hacer creer al servidor que las peticiones provienen de la IP determinada aunque realmente provienen de nuestro PC... pero eso es tema bastante avanzado y antes hay que leer mucho HXC ;)

2. Creación de sitios virtuales

Cambiamos de tema, ahora vamos a crear sitios virtuales, para ello vamos a explicar de manera rápida en que consiste y la razón de crear sitios virtuales.

Cuando contratamos un espacio web a una empresa de hosting ¿qué crees que realmente nos están vendiendo?, nos alquilan un cacho de un gran servidor que tienen, este servidor alojará un número variable de sitios. Es decir, la empresa tendrá un gran disco duro y nos alquilará los megas de ese disco duro según el plan que le contratemos. El microprocesador, la memoria y demás recursos del servidor estarán compartidos por todos los clientes. Si estás pensando en crear un proceso que sature el micro para que todos los dominios caigan, estás en lo cierto, puedes hacer caer todo el sistema si el servidor Apache está mal configurado y dejar varios dominios sin funcionar.

Mediante una correcta configuración del archivo httpd.conf es posible crear nuevos sitios y convertir nuestro servidor dedicado en un potente servidor de hosting ;)



La mayoria de ...

La mayoría de los sitios web son virtuales, muy pocos sitios tienen un servidor dedicado (un servidor 100% dedicado para el sitio web).

A lo largo de estos capítulos hemos configurado nuestro servidor Apache y de momento este servidor es dedicado, es decir, estamos ofreciendo todos los recursos del sistema a un sólo sitio web. ¿Has pensado ofrecer espacio web a tus amigos?, eso es lo que vamos a hacer: Vamos a ofrecer hosting a los amigos para que puedan alojar sus páginas web, mp3, divx en nuestro servidor y que cada uno tenga su propia URL.

En este capítulo vamos a crear sitios virtuales utilizando puertos no estándar, cuando expliquemos como tener un servidor DNS en nuestro propio servidor crearemos sitios virtuales mediante dominios. De momento, quédate que vamos a crear sitios virtuales mediante puertos para que nuestros amigos puedan tener su propio sitio web en nuestro servidor web y que compartan sus archivos.

Vamos a crear 3 sitios virtuales para 3 amigos, que los hemos llamado amigo1, amigo2 y amigo3.



Por favor ...

Por favor, si eso de DNS te suena extraño, es el momento de que te leas el número 1 de HXC, lo tienes a tu disposición en nuestra Web (www.hackxcrack.com) y puedes descargártelo enterito y sin pagar un solo euro.

Si, ya lo se, somos muy pesados, pero recibimos cientos de mails preguntando cosas que ya se ha explicado en números anteriores. Si no pides los números atrasados al menos lee el número 1 antes de enviarnos tus dudas y ya de paso os recordamos que EL FORO es el lugar perfecto para resolver tus dudas, que vía mail es mas lento y no podemos atender todas las peticiones.

Paso 1. Crear directorios

Lo primero es crear los directorios donde los amigos podrán alojar sus archivos y que estos archivos puedan ser accedidos desde Internet.

Creamos una carpeta llamada amigos en el mismo nivel de la carpeta htdocs, en realidad podemos crear esta carpeta donde queramos, incluso en otros discos duros. Ahora creamos las carpetas **amigo1**, **amigo2** y **amigo3** dentro de la carpeta amigos tal y como muestra la imagen.

Paso 2. Configurar Apache

Apache
amigos
amigo1
amigo2
amigo3
bin
cgi-bin
conf
htdocs
icons
include
lib
libexec
logs
modules
proxy

Vamos a ver cómo se puede configurar el servidor Apache para que utilice un único demonio de proceso para trabajar con todos los sitios web virtuales.

Apache utiliza la etiqueta *<VirtualHost>* para definir las propiedades de cada sitio virtual, como ya sabemos esta etiqueta hay que colocarla en el archivo httpd.conf. En primer lugar vamos a configurar de manera básica los tres sitios webs, para ello tenemos que asignar las urls.

La url del amigo1 será ***http://127.0.0.1:8080***
La url del amigo2 será ***http://127.0.0.1:8081***
La url del amigo3 será ***http://127.0.0.1:8082***



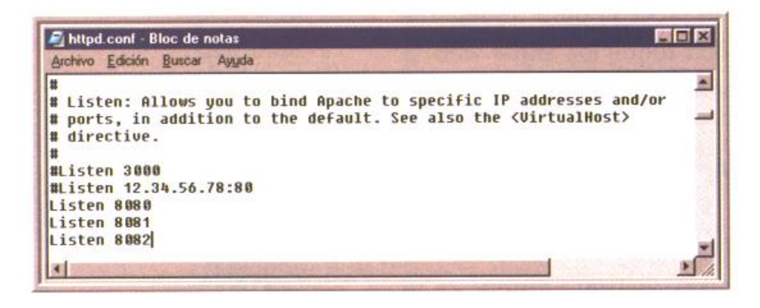
Sustituye ...

Sustituye 127.0.0.1 por la IP de tu conexión, en este ejemplo hemos colocado 127.0.0.1 para realizar las pruebas, pero recuerda que para que tus amigos puedan acceder a sus sitios tendrás que colocar la IP que te ha asignado el proveedor de acceso, eso ya lo explicamos en anteriores entregas.

La IP es la misma, lo único que cambia es el puerto, a cada sitio se le ha asignando un puerto, te recomendamos que sean puertos no estándar (mayores a 1024) de lo contrario puede que ocupes un puerto ya en funcionamiento.

Tenemos que configurar Apache para que esté a la escucha por los puertos 8080, 8081 y 8082, para ello añadimos los siguientes parámetros en el archivo de configuración httpd.conf.

Listen 8080
Listen 8081
Listen 8082



Nos colocamos al final del archivo httpd.conf y añadimos las siguientes lineas:

<VirtualHost 127.0.0.1:8080>
DocumentRoot
"c:/apache/Apache/amigos/amigo1/"
ServerName Amigo1
</VirtualHost>

<VirtualHost 127.0.0.1:8081>
DocumentRoot
"c:/apache/Apache/amigos/amigo2/"
ServerName Amigo2
</VirtualHost>

<VirtualHost 127.0.0.1:8082>
DocumentRoot
"c:/apache/Apache/amigos/amigo3/"
ServerName Amigo3
</VirtualHost>



Las etiquetas ...

Las etiquetas DocumentRoot y ServerName ya fueron explicadas en el capítulo anterior del número 8 de hackxcrack.

Al reiniciar el servidor Apache ya están accesibles las siguientes URLs:

http://127.0.0.1:8080
http://127.0.0.1:8081
http://127.0.0.1:8082

Ahora queda que vuestros amigos puedan subir páginas al servidor y para ello instalaremos un servidor FTP (puedes utilizar el servidor UFTP cuyo funcionamiento ya ha sido explicado en hackxcrack). Crea los 3 usuarios en el Servidor FTP apuntando cada uno a la carpeta específica (amigo 1, 2 y 3) y comunica a tus amigos los datos de acceso (user/pass), de esta forma cada usuario tendrá acceso únicamente a su carpeta y no verá las demás (salvando algunos detalles, esto es lo que hacen las grandes empresas de "hosting" cuando te venden un sitio donde colocar tu Web).

Como puedes comprobar todo lo aprendido en números anteriores te está resultado de gran

utilidad.

3. ¿Qué has aprendido?

Este capítulo ha sido denso y con muchos nuevos conceptos, pero ha merecido la pena, ¿no crees?. Has aprendido el concepto de los módulos de apache, a utilizar los módulos mod_info y mod_status, a obtener la versión de Apache utilizando el netcat, algunos bugs de Apache y a crear sitios virtuales utilizando los puertos.

En el próximo número ....

Se explicarán nuevos módulos, nuevas directivas de configuración para los servidores virtuales y a interpretar los logs que genera Apache, un gran fichero donde se registran los accesos al servidor con datos del navegante (que interesante....)

David C.M

VISUAL BASIC (IV): MI PRIMERA DLL Y ACCEDIENDO A DATOS

POR PEDRO DEL VALLE

Hola de nuevo. Hoy intentaré introduciros en el mundo del acceso a datos, pero antes os explicaré que es una *DLL* y haremos una práctica para entenderlo mejor. Al igual que los *OCX*, explicados en el curso anterior, las *DLL* son herramientas creadas por los programadores para facilitar el trabajo a la hora de desarrollar una aplicación. Dentro de las *DLL* solemos encontrar rutinas y funciones que, o bien son muy complejas, o simplemente se repiten con frecuencia. Pero más importante es aun su función de encapsulado. El programador que utilice tu *DLL* o tu *OCX* no tiene ni debe saber el funcionamiento interno de esta. Pongamos por ejemplo una *DLL* que se encargue del acceso a datos de una BD de Access. Esta *DLL* tiene unas funciones implícitas, que son, por ejemplo, ***Ruta*** (donde indicamos la ruta de la base de datos), ***Nombre*** (nombre de la base de datos), ***Conectar*** (abre la conexión contra la BD) y ***TraerRegistros*** (que nos va a traer todos los registros de la tabla que le indiquemos).

También podríamos implementar otras funciones como ***Añadir, Editar, Borrar***. Y son precisamente estas tres últimas las que dan sentido a nuestra *DLL*, ya que serán utilizadas varias veces en nuestro proyecto.

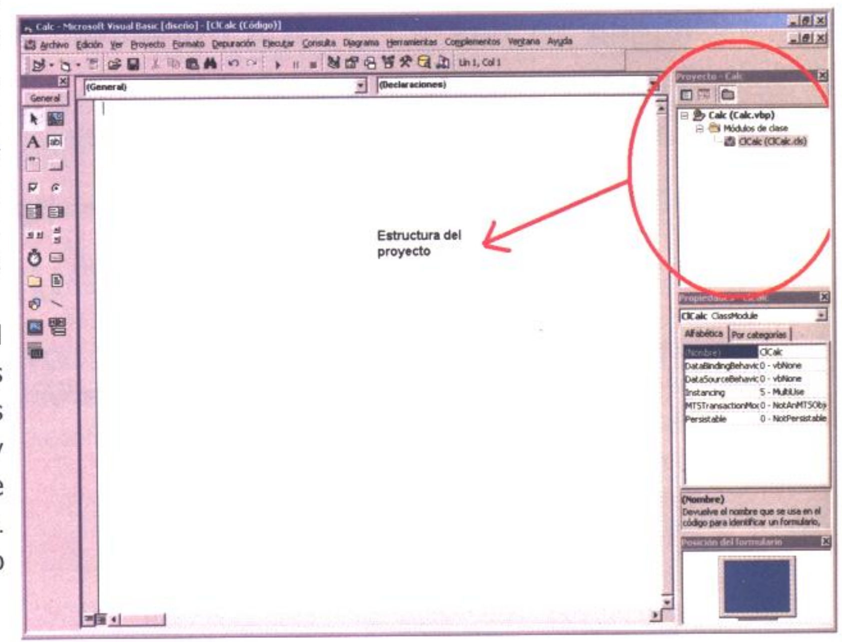
Bien, una vez hecha la introducción a las *DLL*, vamos a crear una desde 0. Abrimos el Visual Basic, y en la primera pantalla vamos a escoger "*DLL* ActiveX".

Vemos que se nos abre el editor, como siempre, pero con la peculiaridad de que a nuestra derecha, en el explorador de proyectos, no existen formularios, sino solo un módulo de tipo clase llamado por defecto ***Class***.

Vale, lo primero que vamos a hacer es darle nombre al proyecto, por lo que nos pondremos encima de ***proyecto1*** en el explorador de proyectos y cambiaremos el nombre en el cuadro de propiedades.

En el ejemplo de hoy, nuestra *DLL* será sencilla, ya que me reservo lo mas complicado para la segunda parte, cuando tratemos el acceso a datos. Esta *DLL* sencillamente realizará las 4 operaciones básicas matemáticas, suma, resta, división y producto. Yo he llamado a mi proyecto **Calc**.

Ahora deberíamos cambiar también el nombre de la clase, para ello vamos al explorador de proyectos, nos posicionamos sobre la ***Class1*** y cambiamos su nombre en el cuadro de propiedades, por ejemplo, ***ClCalc***. Ahora deberíais tener el proyecto como podemos ver en la imagen:



Muy bien, como ya he dicho anteriormente, esta DLL será sencilla, pues solo quiero enseñaros como poder crearlas y su funcionamiento. Lo que vamos a hacer es crear una función por cada una de las operaciones que queremos codificar, es decir, cuatro. El código quedaría así:

```
Public Function Suma(a As Integer, b As Integer) As Integer
   Suma = a + b
End Function

Public Function Resta(a As Integer, b As Integer) As Integer
   Resta = a - b
End Function

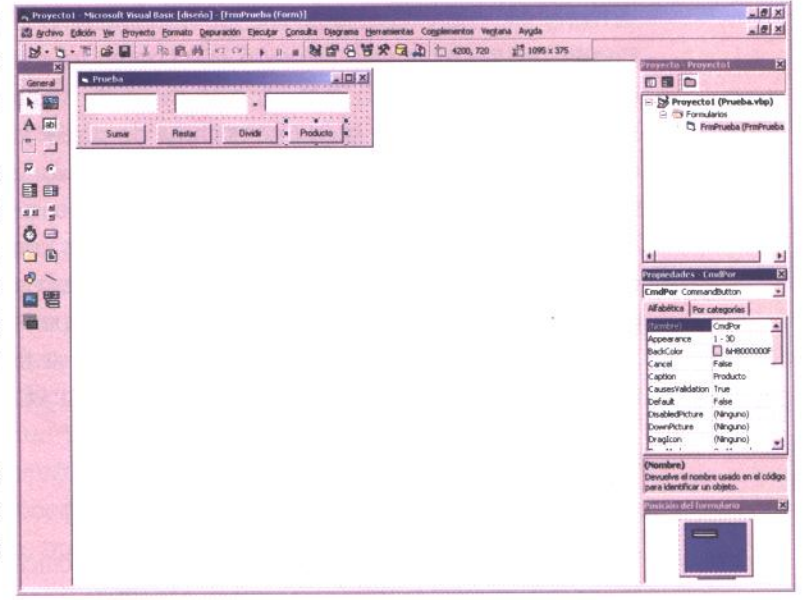
Public Function Producto(a As Integer, b As Integer) As Integer
   Producto = a * b
End Function
Public Function Division(a As Integer, b As Integer) As Integer
   Division = a / b
End Function
```

Cojamos la primera de las funciones. Su nombre es ***Suma***, ya que será la encargada de sumar dos números. Como podemos ver entre paréntesis, la función recibe 2 parámetros (a y b), los cuales son de tipo entero, aunque podrían haber sido también de tipo ***long*** o ***double***. También podemos observar que justamente después del cierre de los paréntesis tenemos escrito ***As Integer***. Esto es debido a que vamos a realizar el cálculo dentro de nuestra función y devolver el resultado, y este ***As Integer*** es el tipo de dato que vamos a devolver.

Lo que sigue es sencillo, estamos diciendo que el valor que devuelve la función es la suma de ***a + b***.

Una vez escritas estas líneas, vamos a generar la DLL. Abrimos el menú Archivo y picamos en la opción ***Generar DLL***. Podemos dejarle el nombre que viene por defecto, que si nos damos cuenta es el mismo que hemos puesto en el nombre de la DLL en nuestro proyecto.

Una vez generada, nos olvidaremos de ella y abriremos un nuevo proyecto, pero esta vez picaremos en ***EXE Estándar***. Ahora, y como siempre, vamos a darle un nombre al ***form*** y al proyecto. Por ejemplo, yo he llamado al proyecto **Prueba** y al Form **FrmPrueba**.



Colocaremos un par de cajas de texto para recoger los valores, otra para el resultado y 4 botones para las operaciones matemáticas.

El diseño del proyecto es totalmente opcional, yo aquí lo he hecho bastante simple, pero podéis poner un **label** entre las dos primeras cajas de texto que cambie conforme pulsas un botón u otro, con el operador correspondiente (+,-,*,/).
¿Todo bien?, perfecto, vamos entonces a registrar la *DLL* que anteriormente hemos generado. Para ello vamos al menú **Proyecto** -> **Referencias**. Podemos ver que hay varias *DLL* seleccionadas por defecto, estas son necesarias para el correcto funcionamiento de los proyectos en VB.

Picamos el botón **Examinar** y buscamos nuestra *DLL*, allá donde la hemos generado, y la agregamos a nuestra aplicación.
Si lo hacemos bien, deberíamos poder declarar una variable del tipo ***Calc*** en mi caso, y en el vuestro, dependiendo del nombre que halláis utilizado anteriormente.
Vamos al editor de código e intentamos declarar una la variable del tipo ***Calc***, ¿cómo?, así:

```
Option Explicit
Dim Cl As Calc.ClCalc
```

Ya tenemos declarada la variable, pero no está instanciada.
Este tipo de variables, sobre todo aquellas que sean objetos o correspondan a *DLL* deben ser instanciadas. Para ello vamos al ***Form_Load()*** y escribimos lo siguiente:

```
Private Sub Form_Load()
   Set Cl = New Calc.ClCalc
End Sub
```

Pasemos ahora a codificar los botones. Vamos al botón Sumar, por ejemplo, y escribimos:

```
Private Sub CmdMas_Click()
   TextRes = Cl.Suma(Texta, Textb)
End Sub
```

Donde ***TextRes*** es la caja de texto que contendrá el resultado y ***Texta***, ***Textb*** son los dos valores que queremos sumar.
Si lo probáis veréis que la *DLL* se encarga de sumar los dos números y devuelve el valor a la caja de texto, aunque también podríamos poner una variable.
El resto de operaciones es exactamente lo mismo, así que os lo dejo a vosotros.
Y con esto, hemos acabado la *DLL*. Como podéis ver, una *DLL* y un *OCX* no son muy diferentes entre si. Realmente, la diferencia más grande que hay entre los dos es que el *OCX* posee entorno grafico para poder interactuar con él, mientras que la *DLL* no.
Suponiendo que habéis entendido el funcionamiento de los *OCX* y *DLL*, creo que ya estamos listos para ir al tópico mas extendido de la programación: **El ACCESO A DATOS**.

ACCESO A DATOS

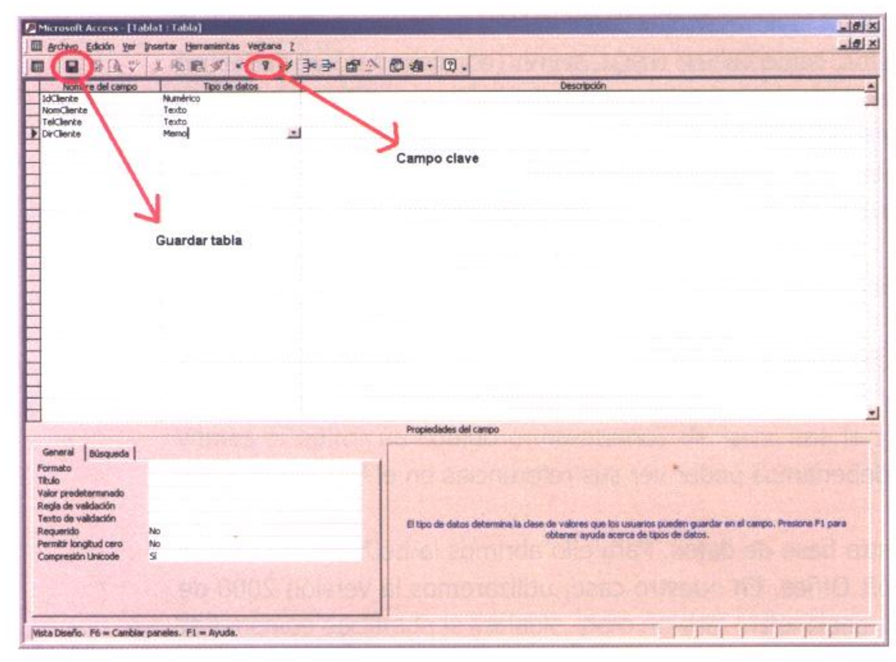
El acceso a datos es, actualmente, una de las técnicas mas solicitadas en el mercado laboral en cuanto al mundo de la programación se refiere, y también es muy necesario para aplicaciones

que necesiten registrar información o datos de cualquier tipo.
Los grandes gestores de base de datos, como **Oracle** o **SQL Server** (e incluso **MySql**) tienen en común un punto muy importante, el lenguaje que interpretan. Este es conocido como **SQL**, y, aunque con algunas diferencias de sintaxis, todos los gestores de base de datos entienden las *querys* generadas en **SQL**, y se pueden crear desde cualquier lenguaje de programación. Ya que no todo el mundo puede tener un **Oracle** en casa, nosotros usaremos una base de datos Access para nuestra práctica.
Bien, empecemos entonces.

Si recordáis, en la primera entrega del curso os indiqué los componentes de acceso a datos que debíais instalaros, para estandarizar el uso de estos. Estos componentes se llaman **MSDAC 2.6 sp1** y **JET**.
Si están correctamente instalados, deberíamos poder ver sus referencias en el Visual Basic, las cuales os indicaré posteriormente.
Antes de nada, debemos crear nuestra base de datos. Para ello abrimos la herramienta Access que viene en el paquete de Microsoft Office. En nuestro caso, utilizaremos la versión 2000 de la misma.

En la primera pantalla picaremos en la opción "**base de datos en blanco**", y la guardaremos con el nombre ***bd.mdb*** en la misma carpeta donde se encontrará nuestro nuevo proyecto. Una vez creada, vamos a la opción "**crear una tabla en vista de diseño**". Cuando estemos dentro veremos que nos aparece un grid con las columnas ***Nombre de campo, Tipos de datos y descripción***. En esta pantalla vamos a diseñar nuestras tablas, que en nuestro caso, para simplificar, solo será una.

Bien, nuestra tabla será un clásico, la tabla "**Clientes**". Debemos empezar a crear los campos. Por ejemplo, el primero, será el identificador de cliente, el cual podemos llamar ***IdCliente***. En "**tipo de dato**", vamos a declararlo ***Numérico***, Ya que contendrá un número que será siempre único. Al indicarle el tipo de dato, vemos que abajo podemos cambiar las propiedades de este, ya que el tipo numérico puede tener muchas variantes (recordemos que los números pueden ser enteros, dobles, long...)
En este caso dejaremos las propiedades que aparecen por defecto, es decir: ***entero largo***, formato en blanco, posiciones decimales Automático, etc...
El siguiente campo podría ser el nombre del cliente, por ejemplo, ***NomCliente***. Este campo será de tipo ***Texto***, y en sus propiedades le diremos que la longitud sea de 100. El siguiente campo contendrá el teléfono del cliente, llamémosle ***TelCliente***. Este es un caso anecdótico, porque a mas de uno seguro que se le ha ocurrido declarar este campo como numérico. Eso es una barbaridad, teniendo en cuenta la cantidad de números que puede llegar a tener un teléfono, e incluso puede que, en algunos casos, se deban insertar símbolos o caracteres no numéricos, como por ejemplo el signo "+", para el "+34" que simboliza el territorio español. Por lo tanto, este será un campo de tipo ***Texto*** con longitud 15.
Por último, añadiremos un campo con la dirección. Le llamaremos ***DirCliente***, para no variar, y será de tipo ***Memo***. Este tipo de campo es poco recomendable, ya que se trata de un tipo Texto pero sin longitud máxima. Por ejemplo, este tipo de campo es utilizado en aquellos foros donde no hay límite d caracteres en los posts.
Si todo ha ido bien deberíais tener el diseño como el de la imagen.

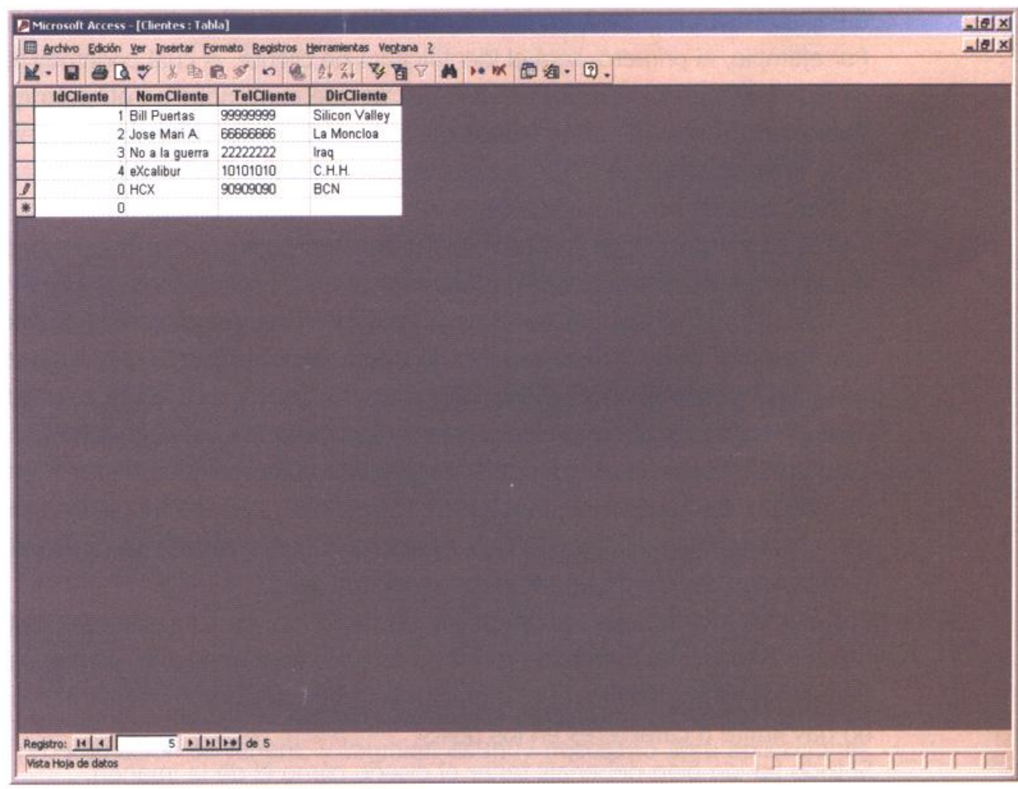


Para acabar, pulsamos en el botón de guardar, e introducimos el nombre de la tabla, si no lo habíais hecho antes.

¿Os ha aparecido un mensaje? ¿Sí?, no pasa nada, es totalmente normal. Este mensaje nos está diciendo que no hemos indicado ninguna clave en el diseño de la tabla. Los campos claves son aquellos cuyo valor no pueden repetirse bajo ningún concepto. Es posible crear claves con mas de un campo, es decir, con 2, 3, 4..., pero en nuestro ejercicio el único campo clave que hay es ***IdClietne***.

Así pues, le decimos que no queremos que nos cree una clave principal, y así le indicamos cual es la que nosotros hemos elegido. Para hacer esto, señalamos toda la línea de ***IdCliente*** y picamos sobre el icono de la llave que podemos encontrar en la barra del menú. Si lo habéis hecho bien, os debería aparecer una pequeña llave en este campo.

Ahora si, picamos de nuevo en guardar y cerramos la ventana hija. En la pantalla que nos aparece deberíamos tener un pequeño icono que simboliza la tabla clientes, y que si hacemos doble click sobre él, nos permitirá introducir datos.

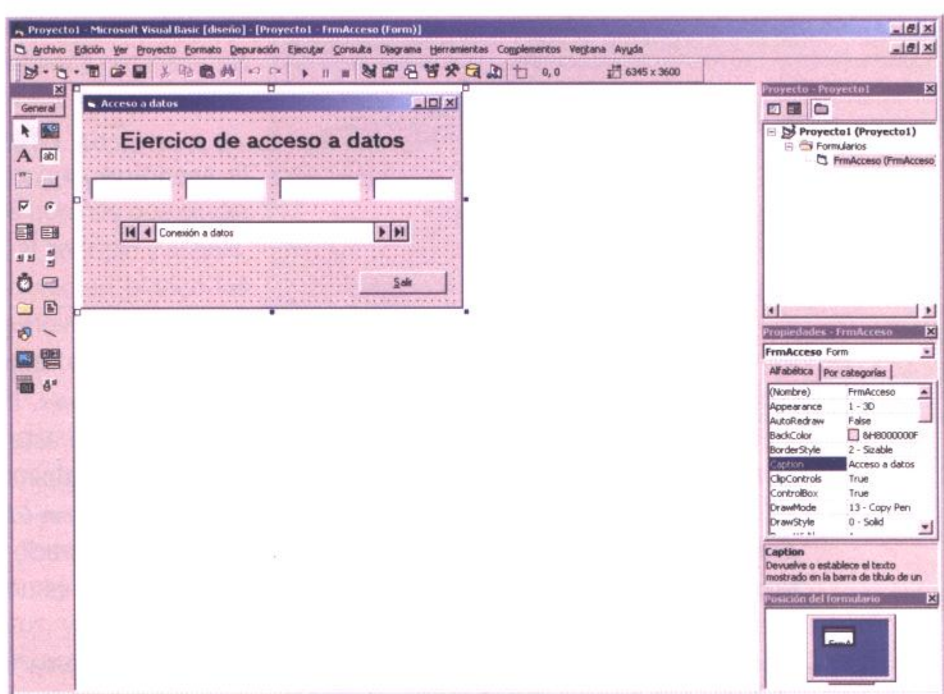


IdCliente	NomCliente	TelCliente	DirCliente
1	Bill Puertas	99999999	Silicon Valley
2	Jose Mari A.	66666666	La Moncloa
3	No a la guerra	22222222	Iraq
4	eXcalibur	10101010	C.H.H.
0	HCX	90909090	BCN
0			

Podemos introducir unos 4 o 5 clientes, inventándonos los datos, pero recordad que ***IdCliente*** es campo clave y por lo tanto es único, por lo que no podréis repetir el número, y si lo hacéis, debería devolveros un error y no permitirlo. Un ejemplo de clientes es el que podéis ver en la imagen.

Una vez introducidos, damos al botón salvar, y salimos por completo del Access.

Vamos a pasar al Visual Basic, para programar una aplicación que sea capaz de leer la tabla y navegar a través de los registros. Es ahora entonces cuando debemos agregar las DLL y los OCX necesarios para el acceso a datos. Abrimos un nuevo proyecto en VB y vamos a referencias. Tenemos que agregar, de la lista, ***Microsoft ActiveX data object 2.6 library***. Todos deberíamos tener esta versión, la **2.6**, ya que instalamos estos componentes en la primera entrega del curso. Es el turno de los OCX. En este caso, solo será necesario agregar el ***Microsoft ADO Data Control 6.0***, el cual nos será muy útil para la introducción de hoy, pero debéis saber que en un futuro debemos intentar prescindir de ellos.



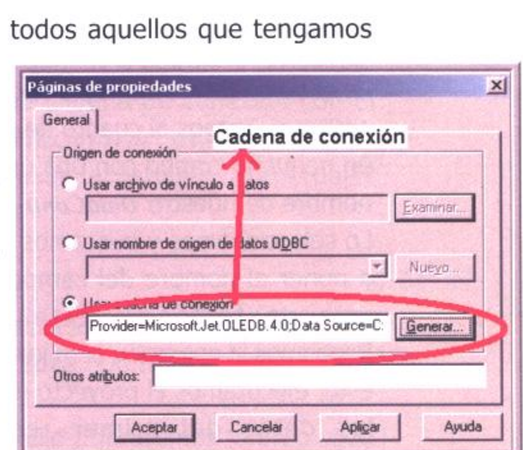
Ahora toca lo mas pesado, diseñar, cambiar nombres, poner objetos...

En nuestro caso, debemos tener 4 cajas de texto (tantas como campos) para visualizar los valores, un ***DataControl***, y un botón salir. Una posible opción sería esta:

Bien, ahora, lo primero que debemos hacer, es abrir una conexión con la base de datos. Para hacer esto, usaremos una cadena de conexión, donde especificaremos el proveedor de datos (Access), la ruta, el nombre de la BD, user y password, si los hubiere.

Es aquí cuando vamos a utilizar un pequeño truco, porque aunque podríamos tener apuntadas las cadenas de conexión de cada gestor, eso sería bastante engorroso. Vamos a nuestro DataControl, que yo he llamado DC (original, ¿no?). Buscamos la propiedad ConnectionString en el cuadro de propiedades y abrimos las opciones, con doble click o pulsando sobre el botón buscar (...). Nos aparece una página de propiedades, donde vemos seleccionado "**Usar cadena de conexión**". Es correcto así, pulsemos entonces el botón "**Generar**".

Vemos que se despliega una lista de proveedores, que serán todos aquellos que tengamos instalados localmente en nuestro PC. El proveedor de Microsoft Access es el Microsoft Jet 4.0, que todos deberíamos tener instalado (si no lo tenéis, probad con la versión anterior). Pulsamos siguiente y nos aparece una pantalla donde debemos pulsar el botón buscar y seleccionar la base de datos de Access, que debería estar en la misma carpeta que donde vais a guardar este proyecto. Una vez seleccionada, pulsad "**Probar conexión**", para verificar que lo hemos hecho bien.



¿Ha ido bien?, aceptad entonces y copiar en el portapapeles la cadena generada, borrándola posteriormente de donde está, ya que nosotros vamos a conectar por código.

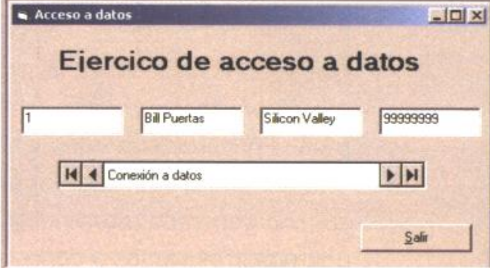
Vamos al evento Form_Load() del formulario, y creamos la conexión. Para ello debemos indicarle a la propiedad ConnectionString del DataControl la cadena de conexión que hemos guardado en el portapapeles. Una vez puesto esto, vamos a depurarlo mas, y allá donde pone la ruta, vamos a utilizar un objeto de VB para indicársela, porque si no, esto solo funciona cuando la base de datos se encuentra en él la ruta especificada en la cadena de conexión. Este objeto que vamos a utilizar, se llama App, y la propiedad que nos devuelve la ruta es la .Path. Entonces, la cosa quedaría así:

```
Private Sub Form_Load()
    DC.ConnectionString = "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=" & App.Path &
                          "\bd.mdb;Persist Security Info=False"
   DC.RecordSource = "Clientes"
   DC.Refresh
End Sub
```

Vemos que se ha sustituido la cadena por el objeto *App.Path*. También podemos observar que a la propiedad *RecordSource* le hemos pasado el literal *Clientes*. En la propiedad *Recordsource* indicamos la sentencia *SQL* que queremos que se ejecute. En este caso, y como queremos que se traiga todos los registros de la tabla *Clientes*, podemos escribir
DC.RecordSource = "Clientes" **o** *DC.RecordSource = "Select * from Clientes"*, que tendría el mismo significado. En breve haré una propuesta en los foros de hackxcrack, para saber si creéis necesario una entrega completa explicando la sintaxis de *SQL*, ya que es un mundo realmente complejo y extenso.
Bien, por último podemos ver escrito *Dc.refresh*. De esta manera "refrescamos" o actualizamos el *DtControl*, y es necesario, ya que si no lo ponemos jamás conectaría con la base de datos.
Para el botón Salir, lo único que debemos hacer es cerrar la conexión y cerrar el formulario. Esto se haría así:

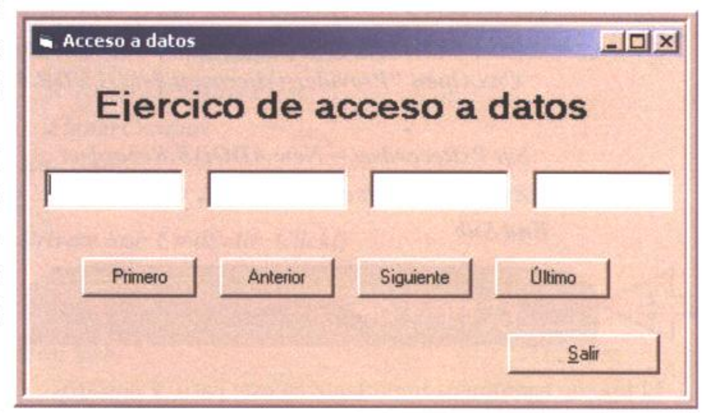
```
Private Sub CmdSalir_Click()
   DC.Recordset.Close
   Unload Me
End Sub
```

Vemos que lo que estamos cerrando realmente es el *Recordset*, y eso es debido a que, cuando nosotros cargamos el *Recordsource*, realmente estamos llenando el *Recordset* que lleva intrínseco el *Datacontrol*.
Es el turno de las cajas de texto. Pongamos el foco sobre el primer *TextBox*, el que mostrará el *IdCliente*. Vamos al cuadro de propiedades y buscamos *DataSource*. Vemos que se nos despliega un pequeño combo con una única opción, que es precisamente el nombre de nuestro *DataControl*.
Lo seleccionamos y buscamos la propiedad *DataField*. Aquí vamos a poner el nombre del campo que queremos que muestre esta caja de texto. En este caso, escribiremos *idCliente*. Repetimos la operación para los restantes *TextBox*. Una vez acabado esto, ejecutamos el proyecto y... ¡Ualap!, deberíamos estar viendo los datos del primer registro en nuestro formulario.



Si pulsamos las flechas del *DataControl* deberíamos poder navegar, sirviendo unas para moverse paso a paso y otras para ir al inicio o al final de los registros de la tabla. Esta es la manera simple y, recordemos, poco recomendada, de acceder a datos. Pero claro, para este ejercicio tan sencillo, dentro de lo que es el mundo de acceso a datos, he preferido enseñaros primero este camino, el fácil, y ahora explicaros otra forma mucho más correcta de hacer exactamente lo mismo. Esta otra manera será implícitamente por código, y no por diseño. Los *TextBox* volverán a tener en blanco sus propiedades *DataSource* y *DataField* y eliminaremos el *DataControl* para dar paso a los objetos *Connection* y *Recordset* de *ADO*.

¿Estáis listos?, bien, eliminemos entonces el *DataControl* y añadamos unos botones que sirvan para moverse adelante, atrás, al principio y al final. También no os olvidéis de dejar en blanco las propiedades que habíamos cambiado de las cajas de texto. Quedaría algo como esto:



Vamos ahora a la declaración de variables. Hoy conoceremos dos nuevos tipos de dato, muy importantes para la conexión y gestión de datos. Estos objetos son el *ADODB.Connection* y el *ADODB.Recordset*

El primero, es el encargado de realizar la conexión contra la base de datos, ya sea un *Access*, *Oracle* o *SQL Server*. A este objeto debemos indicarle, entre otras cosas, el proveedor y la localización de la base de datos, ya sea por ruta o por *ODBC*.

El segundo, *Recordset*, no es mas que un puntero (concepto que mejor no explicaré, pues tendríamos para 10 capítulos) que apunta a la base de datos. Este objeto es el que contendrá los registros que nosotros queramos, indicándoselo en su propiedad open. Por ejemplo, un recordset puede apuntar a todos los registros de la tabla clientes, o solo a aquellos clientes que se llamen "*Camilo*", o los que vivan en "*Barcelona*".

Después de esta explicación técnica, vamos a la práctica. Declaremos las dos variables anteriormente mencionadas. Por ejemplo:

```
Option Explicit
Dim Cnx As ADODB.Connection
Dim RsRecordset As ADODB.Recordset
```

Vamos ahora al *Form_Load()* y generemos la conexión. Para ello debemos utilizar la propiedad *Open* de nuestra variable *Connection*. Tenemos que "pasarle" la cadena de conexión con el proveedor y la ruta, tal y como hicimos con el *DataControl* anteriormente. Casi se me olvida deciros que, anteriormente, debéis instanciar la variable, ¿recordáis?

```
Set Cnx = New ADODB.Connection
```

Así es como la instanciaríamos. Con estas dos líneas, hemos realizado una conexión contra la base de datos, pero aún no nos hemos traído el conjunto de registros que necesitamos, y para ello pediremos ayuda a nuestro nuevo amigo, el *Recordset*.

Justo después de abrir la conexión, abriremos el *recordset*. Para ello utilizaremos la propiedad

.Open, (Recordad instanciar también este objeto) y le indicaremos el nombre de la tabla, la conexión y el modo de apertura y bloqueo del ***recordset***.
Los modos de apertura indican si queremos el ***recordset*** en modo lectura, escritura, y el bloqueo indica que pueden hacer otros usuarios con este registro cuando nosotros lo estamos leyendo o editando. Bueno, nuestro ***Form_Load()*** quedaría así:

```
Private Sub Form_Load()
   Set Cnx = New ADODB.Connection
   Cnx.Open "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=" & App.Path & "\bd.mdb;Persist
                                                                    Security Info=False"
  Set RsRecordset = New ADODB.Recordset
  RsRecordset.Open "Clientes", Cnx, adOpenDynamic, adLockOptimistic
End Sub
```



El tipo de...

El tipo de apertura es importante en este caso, y nosotros lo abrimos de forma Dinámica, que permite actualizaciones. En cambio el tipo de bloqueo, nos es poco importante ya que solo trabajaremos nosotros contra esa base de datos.

Bien, ya hemos abierto una conexión y recogido los registros de la tabla Clientes. Ahora deberíamos mostrar los datos en los ***TextBox***. Para ello podemos añadir debajo del ***RsRecordset.Open*** las líneas necesarias. El ***Recordset*** contiene los datos, y para acceder a ellos lo haremos de la siguiente forma:

```
TxtIdCliente = RsRecordset("IdCliente")
TxtNomCliente = RsRecordset("NomCliente")
TxtDirCliente = RsRecordset("DirCliente")
TxtTelCliente = RsRecordset("TelCliente")
```

Si os fijáis, entre los paréntesis vemos el literal con el nombre del campo, pero también podríamos referirnos a estos con su índice, por ejemplo, ***RsRecordset("NomCliente")*** sería ***RsRecordset(2)***.
Ejecutamos la aplicación y vemos que las cajas de texto se han rellenado con los datos de la BD. Ahora deberíamos movernos por los registros.
Vamos al botón siguiente, y en su evento ***Click*** pondremos ***RsRecordset.MoveNext***. Con esto nos moveremos al siguiente registro, pero debemos volver a cargar las cajas de texto con los datos. Ya que esto lo vamos a tener que hacer en cada botón, mejor creamos una función que mueva los valores del recordset a los ***TextBox***. La función la podríamos llamar ***LlenarCampos***, y la codificaríamos así:

```
Public Function LlenarCampos()
   TxtIdCliente = RsRecordset("IdCliente")
   TxtNomCliente = RsRecordset("NomCliente")
   TxtDirCliente = RsRecordset("DirCliente")
   TxtTelCliente = RsRecordset("TelCliente")
End Function
```

Una vez escrita la función, el botón siguiente quedaría así:

```
Private Sub CmdSiguiente_Click()
   RsRecordset.MoveNext
   LlenarCampos
End Sub
```

Los demás, no tiene ningún secreto, podríais intentar hacerlos vosotros mismos, yo os pondré aquí el código, pero antes de leerlo, intentadlo.

```
Private Sub CmdAnterior_Click()
   RsRecordset.MovePrevious
  LlenarCampos
End Sub

Private Sub CmdPrimero_Click()
   RsRecordset.MoveFirst
   LlenarCampos
End Sub

Private Sub CmdSiguiente_Click()
   RsRecordset.MoveNext
   LlenarCampos
End Sub

Private Sub CmdUltimo_Click()
   RsRecordset.MoveLast
   LlenarCampos
End Sub
```

Y con esto, hemos acabado por hoy. Para que practiquéis, codificar el botón salir, y si os dais cuenta, cuando hacéis "siguiente" varias veces, hasta llegar al último registro, el programa devuelve un error, porque intentáis avanzar donde ya no existen datos. Solo os daré una pista, las propiedades ***EOF*** y ***BOF*** indican el final y principio de un recordset respectivamente, devolviendo ***True*** o ***False***.
Un saludo a todos

En la próxima entrega, altas, modificaciones y bajas!!!

Fuente:

```
Option Explicit
Dim Cnx As ADODB.Connection
Dim RsRecordset As ADODB.Recordset

Private Sub CmdAnterior_Click()
   RsRecordset.MovePrevious
   LlenarCampos
End Sub

Private Sub CmdPrimero_Click()
   RsRecordset.MoveFirst
   LlenarCampos
End Sub

Private Sub CmdSalir_Click()
   RsRecordset.Close
   Unload Me
End Sub

Private Sub CmdSiguiente_Click()
   RsRecordset.MoveNext
   LlenarCampos
End Sub

Private Sub CmdUltimo_Click()
   RsRecordset.MoveLast
   LlenarCampos
End Sub

Private Sub Form_Load()
   Set Cnx = New ADODB.Connection Cnx.Open
"Provider=Microsoft.Jet.OLEDB.4.0;Data
Source=" & App.Path & "\bd.mdb;Persist Security
Info=False"

Set RsRecordset = New ADODB.Recordset
RsRecordset.Open "Clientes", Cnx,
adOpenDynamic, adLockOptimistic

TxtIdCliente = RsRecordset("IdCliente")
TxtNomCliente = RsRecordset("NomCliente")
TxtDirCliente = RsRecordset("DirCliente")
TxtTelCliente = RsRecordset("TelCliente")

End Sub

Public Function LlenarCampos()
   TxtIdCliente = RsRecordset("IdCliente")
```

```
    TxtNomCliente = RsRecordset("NomCliente")
    TxtDirCliente = RsRecordset("DirCliente")
    TxtTelCliente = RsRecordset("TelCliente")
End Function
```

TECNICAS DE PORT SCANNING Y USO DEL NMAP.

1- ¿Qué es el escaneo de puertos y para qué sirve?

El escaneo de puertos es una técnica usada por **hackers** y **administradores** (sin ánimo de hacer distinciones) para **auditar máquinas y redes** con el fin de saber que puertos están abiertos o cerrados, los servicios que son ofrecidos, **chequear la existencia de un firewall** así como verificaciones sobre el funcionamiento del mismo y algunas otras cosas. Ni que decir tiene que ésta es una de las técnicas más utilizadas a la hora de penetrar en un sistema y realizar un análisis preliminar del sistema... sin duda una de las mejores y más efectivas para llevar a cabo nuestras "**oscuras intenciones**".

De eso te hablaré aquí, de las diversas técnicas de escaneo, de cómo escanear una máquina o red y de cómo aprovechar los resultados obtenidos para atacar o proteger nuestro sistema, todo con un propósito puramente educativo ;-)

2- Algunos tipos de escaneo conocidos.

Antes de seguir con la explicación deberías tener algunas nociones básicas acerca del protocolo **TCP** y algunas otras cosas que ya se han explicado en la revista.



Si no estas...

Si no estás iniciado en el tema, lo que leerás a continuación quizás te amedrente un poco. Vamos a ver, este artículo servirá de plataforma para explicar en profundidad muchos temas en próximos números, todo aquello que ahora no entiendas será explicado, como hacemos siempre, pero necesitamos una referencia y este artículo cumplirá esa función.

Algunos tipos de escaneo explicados de forma breve, y que luego verás mejor con la práctica, son:

-TCP connect: esta es una técnica rápida y simple, pero tiene el inconveniente de que canta un poquito xD y se detecta fácilmente. Además por lo general si utilizas esta técnica tus conexiones serán logueadas y/o filtradas.

Se basa en intentar establecer una conexión con el puerto del host remoto mediante la llamada a **connect ()** si se establece dicha conexión, evidentemente, el puerto está abierto.

-TCP SYN: se trata de un escaneo en el que no se establece una conexión completa, enviamos **SYN** y en función de la respuesta obtenida por el host contestamos con **RST** (en caso de estar abierto) para resetear la conexión, es decir, abortar.
Puede darse el caso de que al enviar un paquete **TCP** con el **bit SYN** no se reciba respuesta lo que significa que el host está desconectado o se filtra la conexión a ese puerto.
A este tipo de escaneo se le conoce como **escaneo "medio abierto"** o **"SYN stealth"**.

-Stealth Scan (TCP FIN): se trata de enviar **FIN** y esperar la respuesta del host víctima de nuestro escaneo **FIN stealth ("sigiloso")**, si ignora los paquetes enviados entonces el puerto está abierto. Los sistemas de la compañía **Micro$oft** (entre otros) no son susceptibles a este tipo de escaneo aunque parezca mentira :P .

-Reverse Ident (TCP): realizamos un escaneo normal **TCP** pero miramos si el puerto **113** está abierto con el objetivo de saber quién es dueño de los servicios que corren en otros puertos de la máquina.

-Ping Scan: ...bastante explícito xD en todo caso se debe usar cuando tu intención sea saber que máquina(s) están despiertas, es posible bloquear los pings, pero luego (con la practica) veremos como saltarse esta "protección" en caso de encontrarnos con el inconveniente.

-Bounce Attack (vía ftp): se trata de aprovechar la conexión proxy ftp para escanear a través de un servidor **FTP**. Esto es así porque podemos utilizar el comando **PORT** indicando una dirección **IP** y un puerto, en este caso el objetivo de nuestro escaneo y solicitamos una transmisión de datos, si el puerto en cuestión está cerrado se generará un **error 425** ("Can´t get data conection" o algo muy similar). Es una buena idea deshabilitar la opción **ftp proxy** para evitar que terceros utilicen nuestro servidor para escanear y/o atacar otras redes.

-UDP Scan: este escaneo mostrará los puertos abiertos que utilizan el protocolo **UDP** (con sus inconvenientes), es bastante lento aunque irá mejor si escaneas una máquina que utilice la plataforma **Window$** gracias a la política de **M$** de hacer las cosas "**iguales pero diferentes**" y "**viva el monopolio**".

-ACK Scan: muchas veces nos encontramos con un **bonito firewal**l que impide el "correcto **flujo de los paquetes**" xDD desde nuestra máquina al **host víctima**, por eso y otros motivos nos interesa saber qué tipo de configuración tiene el cortafuegos, es decir si el tráfico ha sido filtrado o no.

-Window Scan: muy parecido al anterior, pero nos dice también que puertos están abiertos.

-Null Scan: se trata de otro método de escaneo stealth, en el que enviamos un curioso paquete sin banderas levantadas.

-Xmas Scan: lo realizamos enviando paquetes **TCP** anormalmente configurados y todos los **flags** (banderas) **SYN, ACK, PSH, RST, URG y FIN** levantados.

-Idle scan: se trata de una técnica de **escaneo stealth** muy potente y eficaz, con la que no tenemos necesidad de enviar ni un solo paquete con nuestra IP si no que se utilizan **host's zombis**, sería interesante comentar esta técnica detalladamente en su propio espacio por lo que no profundizaremos ahora.

Finalmente,

-RCP Scan: se trata de enviar el comando **NULL (SunRPC)** a los puertos **tcp** o **udp** que están abiertos y ver si son puertos **RPC** para saber qué programa y su versión está corriendo.

Estos son los tipos de escaneo fundamentales aunque no son los únicos.

3- Nuestros enemigos: "IDS".

Antes de proceder a ver de qué manera podemos aplicar los diferentes ataques existentes que acabamos de repasar brevemente me gustaría dejar clara una cosa a la hora de escanear puertos a diestro y siniestro: **DEBES SER CUIDADOSO Y NO DAR LA NOTA** (--ino j*d*s!) ten en cuenta que aparte de haber buenos administradores (pocos pero hay :P) monitoreando el tráfico y la actividad de sistema, los cortafuegos y los logs del sistema... existen los llamados: **IDS** (Intrusion Detection System, que no traduciré porque seguro que ya sabéis lo que significa xD) mediante estos sistemas es posible detectar un escaneo, activar las alarmas y llevar a cabo las acciones oportunas, es decir, que si un "**UIA**" (Usuario del Intesné

Aburrido) se dedica a escanear ciento y pico máquinas de una red de arriba a abajo en plan "destroyer" tiene todos los números de meterse en un buen lío ...aunque os pego aquí un texto sacado de la web de los *Men In Green* ;-)

<< Es de destacar que conductas tan frecuentes en esta Sociedad de la Información, como son el Spam o el simple Escaneo de puertos, difícilmente encuentran cabida entre los delitos tipificados en nuestro Código Penal, por lo que no son perseguibles por vía penal. >>

Ejemplos de aplicaciones "**IDS**" son: **Cyber Cop, CISCO NetRanger, TripWire, Snort y L.I.D.S** siendo estos tres últimos gratuitos :)



IDS ¿Qué? ...

IDS ¿Qué? ¿?¿? Si nunca has oído hablar de esto, ya sabes, *www.google.com* e investiga un poco por tu cuenta.

4- Visión práctica: usando el NMap.

NMap es una herramienta **LIBRE** para la auditoria de redes, disponible en varias plataformas aunque fue desarrollada inicialmente para entornos **Unix**, y para mí una de las mejores utilidades que existen actualmente para el propósito, mediante esta potente herramienta podemos por ejemplo:

-Determinar que host's están disponibles en una red.
-Determinar los puertos abiertos que tiene el sistema y que servicios ofrecen.
-Determinar que sistema operativo corre en el host objetivo.
-Determinar la configuración y uso de cortafuegos.
-Arreglar una tarde aburrida escaneando el ordenador de nuestra vecina ;)

¿De dónde me bajo el NMap?

Como acabo de decir se trata de una herramienta **LIBRE** y eso significa que dispones del código fuente y el programa compilado; lo tienes para diversas plataformas. Aunque **NMap** está pensado para su uso en la consola (línea de comando) dispone de un agradable **GUI** (Interfaz Gráfica de Usuario) que hace más fácil todavía su manejo, si piensas utilizar **NMap** bajo la plataforma **Window$** te recomiendo que sea un **Window$ NT/2K** aunque existe para **95/98/ME**, por supuesto las versiones para **Window$** no son tan rápidas ni estables como cabría esperar debido a que todavía se encuentran en constante desarrollo.

Te puedes bajar la versión **NMap 1.3.1** para **Window$** aquí:

ttp://download.insecure.org/nmap/dist/nmapwin_1.3.1.exe

INSTALACIÓN DEL NMAP:

Si has optado por descargarte **NMAP** junto con su **GUI ("NMapWin")** simplemente sigue estos pasos:

1-Bájate **NMapwin_1.3.1.exe** (instalador para **Window$**) mencionado justo arriba.

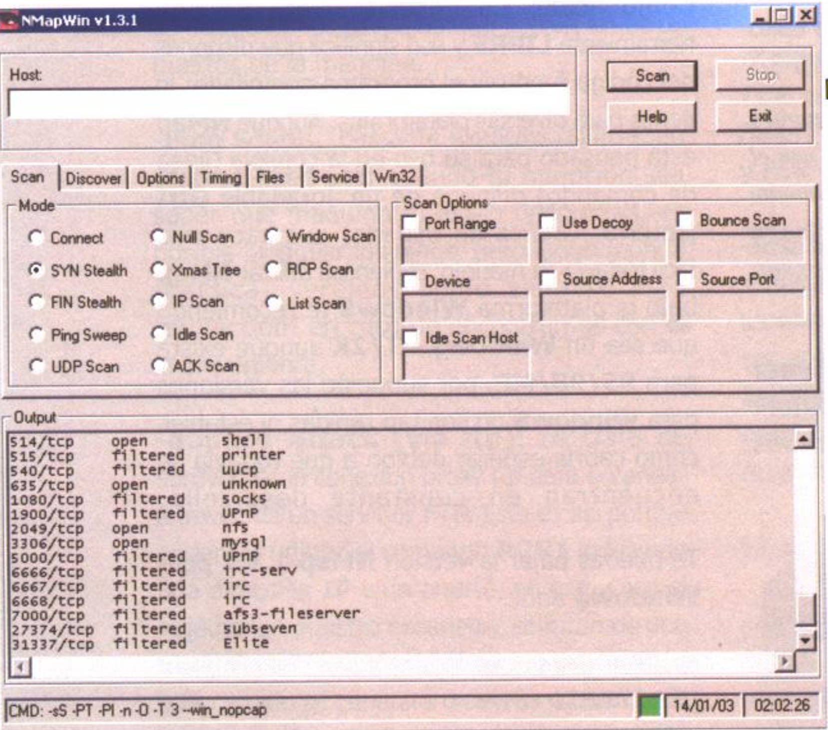
2-Ejecuta el instalador y sigue las instrucciones indicadas.

3-Una vez finalizado esto te vas al directorio que diste al programa y encontrarás nuevos ejecutables, debes instalar **WinPCap**.

4-Una vez reiniciado **Window$** (por una vez que sea decisión tuya ¿no? xD) ya dispones de **Nmap** para la línea de comandos y su interfaz gráfica plenamente funcionales :)

Ahora explicaremos cómo utilizar el **NMap** mediante su **GUI** para escanear los puertos de

un sistema mediante las diversas técnicas que he comentado.



Para empezar lo mejor será ver todas esas opciones que puedes utilizar jeje seguro que estás inquieto pensando en darle al botoncito de "scan" pero no te impacientes.

En primer lugar tenemos un cuadro de texto donde indicaremos el host que queremos escanear, podemos usar la IP o el nombre del host indicando además:

/32 para escanear únicamente el servidor especificado.
/16 para escanear una dirección de clase '**B**'.
/24 para escanear una dirección de clase '**C**'.

Si no entiendes ...

.Si no entiendes eso de Clase A, B, C, no te preocupes, en los próximos números y tomando de referencia este texto explicaremos las MUCHAS cosas que seguro no comprendes ahora mismo de este artículo... poco a poco y paso a paso :)

También se permite el uso del asterisco, por ejemplo: 198.154.3.* e incluso el escaneo de puertos dentro de un rango específico.

Si has leido ...

Si has leído los números anteriores de esta publicación ya sabes lo que es una IP y el formato que tiene, por lo tanto solo te puntualizo que cuando sustituyes una parte de la IP por un asterisco, lo que haces es escanear un rango de IPs.

Si tomamos como ejemplo 198.154.3., lo que hacemos es escanear desde la IP 198.154.3.0 a la IP 198.154.3.255*

Justo abajo tenemos una serie de pestañitas para indicar el tipo de escaneo a realizar así como las opciones adicionales que queremos aplicar. Los resultados obtenidos tras el escaneo aparecerán en el cuadro de salida (Output), además si te fijas en la barra de abajo aparecen los argumentos que deberíamos pasarle al **NMa**p mediante la línea de comandos.

Veamos esas opciones que es importante conocer, por ejemplo ya hemos comentado que es posible que nuestra víctima (por llamarla de alguna manera :P) esté on-line pero no deja que "pinguées" su servidor.

Si te fijas en la pestañita "**SCAN**" verás que puedes seleccionar las modalidades de escaneo que te he explicado hace un momento, y que no volveremos a comentar ahora, pero lo que si que es interesante explicar son las opciones de tu derecha ("**Scan Options**").

Como se puedes ver si quieres indicar un rango de puertos a escanear (con -P) debes marcar la opción **Port Range**, por ejemplo así: **10-2048**.
Otra opción curiosa que me gusta bastante (aunque yo no la uso :P) es **Decoy**, sirve para escanear usando uno o más señuelos para que el servidor objetivo vea que le atacan desde varias **IP**'s que debes indicar separadas por comas, si no indicas "**ME**" en una de las posiciones se te asignará una de oficio, digo, aleatoria xD imagínate que locura para un administrador ;)

Más cosas, en **Device** (-e) especificamos la interfaz desde la que enviar y recibir los paquetes, aunque normalmente **NMap** detecta esto de forma automática :) así que salvo casos puntuales no debes preocuparte por eso.
Otra de las opciones de especial interés es indicar una **IP** en **Source Address** (-S) de manera que aparentemente el ataque se realiza desde esa IP, aunque evidentemente el escaneo no te servirá de nada :(especifica la interfaz en este caso (-e) y ten en cuenta que falsear la dirección de origen por otra ajena puede hacer que "esa IP" se meta en problemas, por favor, utiliza esta función con responsabilidad.

Otra opción es **Bounce Attack** (-b) necesita que le pases usuario:password@servidor:puerto y ya sabes lo que hace ;) pero recuerda que va muy lento.
También podemos especificar el **Source Port** (puerto de origen) con -g o marcando su casilla en la interfaz gráfica con el fin de establecer ese puerto como fuente de nuestro escaneo. Esto tiene sentido especialmente si un **firewall** hace excepciones en función del puerto de origen para filtrar o no los paquetes.

En la pestañita "**DISCOVER**" encontrarás nuevas posibilidades:

-No realizar un ping previo. (-P0)
-Realizar un "ping" TCP usando ACK tal que así: -PT80 (puerto 80 por no ser normalmente filtrado) si el objetivo no permite pings puedes estas dos últimas opciones... aunque es una sugerencia xD.
-Realizar un "ping" TCP usando SYN (-PS).
-Realizar un ping mediante ICMP.
Por defecto NMap realiza un ping ICMP + ACK en paralelo, es decir, -PI y -PT.

Si ahora te vas a "OPTIONS" podrás ver unas opciones muy interesantes, por ejemplo:

Fragmentation (-f) con esta opción se usan paquetes fragmentados que dificultan la detección del escaneo, en un futuro os hablaremos de ataques de ataques de fragmentación y relacionados ampliamente.

OS detection: pues eso, si quieres determinar que tipo de SO utiliza el objetivo marca esta casilla, recomendado.

Get Ident Info: usa la técnica del escaneo inverso (TCP reverse ident scanning) siempre que sea posible, úsalo al realizar un escaneo tipo -sT (connect), útil cuando quieras saber si un servicio corre con privilegios de administrador.

Fast Scan (-F): escanea los puertos especificados de /etc/services (nmap-services en nuestro caso xD).

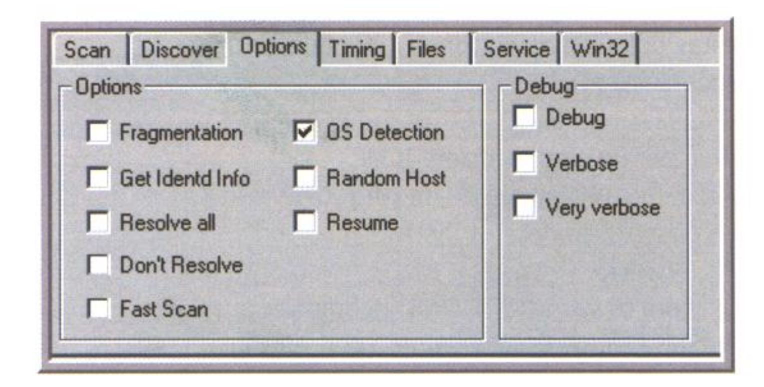
Random Host's: objetivos aleatorios (-iR) y bueno, realmente no nos interesa.

Con don´t resolve (-n) y resolve all (-R) indicamos si queremos resolución DNS, es decir, resolución de nombres de host por su IP.

Mediant Resume (--resume fichero) podemos retomar un escaneo detenido usando un fichero de salida.

Verbose y very verbose: nos da información más amplia de lo que sucede (-v) puedes usarlo dos veces (very verbose) para mostrar todavía

más información.



Si ahora pasas a la pestañita "timing" encontrarás opciones referentes al tiempo que, por lo general, no es necesario que modifiques aunque si por ejemplo te encuentras en una red con tráfico excesivo u otros casos en los que sea necesario reajustar estos valores debes hacerlo aquí, veamos lo que podemos hacer:

Throttle: sirve para configurar el tiempo de envío y respuesta de paquetes de forma general, siendo Paranoid el más lento e Insane el más rápido. (-T Paranoid|Sneaky|Polite|Normal...).

Otras alternativas disponibles (¡no debes usarlas si modificas algo usando Throttle!)

Host Timeout: indica el tiempo máximo en milisegundos que se dedica al escaneo de un único sistema, por defecto no tiene impuesto un límite tal y como se puede ver.

Max RTT: tiempo máximo de espera en milisegundos que NMap esperará a recibir respuesta del sistema escaneado.

Min RTT (ms): para evitar casos de demora en la comunicación NMap nos asegura un tiempo mínimo de espera.

Initial RTT (ms): al especificar la opción -P0 para probar un cortafuegos podemos especificar un tiempo de espera en las pruebas iniciales.

Parallelism: (--max_parallelism num) indica el número máximo de escaneos que se permitirán ejecutar en paralelo, no debes utilizar un valor muy alto para no agotar los recursos de tu máquina ;-)

Scan Delay: tiempo mínimo de espera entre las diferentes pruebas en milisegundos, lo puedes usar para evitar (¿?) que algunos IDS detecten el escaneo.

Pasemos a la siguiente pestañita como ves hay MUCHAS opciones xD para "curiosear" redes; en Files encontrarás Input File y Output File (-iL y -oN respectivamente):

Input File: escanear los objetivos de un fichero dado con objetivos separados por retornos de carro (intro), espacios y tabulaciones.

Output File: guarda los resultados obtenidos en un fichero especificado, podemos especificar el formato del salida pero lo suyo es dejarlo normal (-oN) puedes marcar Append para añadir la nueva información al final del fichero.

A continuación se puede ver la pestaña "Service" que simplemente se usa para configurar el servicio pero para lo que estamos tratando ahora no es importante así que pasemos a otras opciones que aunque tampoco son especialmente necesarias veremos algunas, son las de la pestañita "Win32" y se usan para lo siguiente:

No Pcap (--win_nopcap): deshabilita el soporte Wincap.

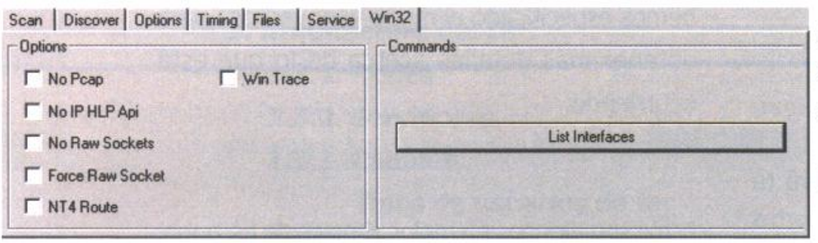
No Raw Sockets (--win_norawsock): deshabilita el soporte para raw sockets (algún día hablaremos de los sockets y los sockets en crudo puesto que proporcionan ventajas interesantes aunque por ahora puede ser demasiado avanzado).

List Interfaces (--win_list_interfaces): lista todas las interfaces de red.

C:\>nmap --win_list_interfaces
Available interfaces:

```
Name        Raw send   Raw recieve  IP
loopback0   SOCK_RAW   SOCK_RAW     127.0.0.1
eth0        SOCK_RAW   winpcap      0.0.0.0
ppp0        SOCK_RAW   SOCK_RAW     211.98.199.104
```

Force Raw Socket(--Win_forcerawsock): probar los raw sockets incluso en un sistema que no sea Window$ 2000.



Hemos visto lo podemos hacer y supongo que te das cuenta de la potente herramienta que tienes en tus manos ¿verdad? jeje pues espera a usarla ;).

4.1- Algunos ejemplos de uso.

Ahora veremos algunos ejemplos de escaneo utilizando el NMap, yo como soy un maniático usaré la línea de comandos, cada uno que use lo que más le guste :)

Antes de ponernos a "curiosear" permíteme recomendarte no escanear desde tu propio PC =:) búscate el cyber cutre de turno o asegúrate de ocultar bien tu IP, etc.
Realizamos un primer chequeo:

Hoy es un día aburrido y no hay nada mejor que hacer que buscar WinGates (servicio ofrecido normalmente en el puerto 1080), abrimos una ventana de comandos y escribimos: (**sustituye "objetivo" por la IP de la víctima**)

nmap -sP -vv -p 1080 **objetivo**

y obtendremos como respuesta algo parecido a esto:

Starting nmap V. 3.00 (www.insecure.org/nmap)
*Host (***.***.**.*) appears to be up ... good.*
*Initiating SYN Stealth Scan against (***.***.**.*)*
The SYN Stealth Scan took 0 seconds to scan 1 ports.
*Interesting ports on (***.***.**.*):*
Port State Service
1080/tcp filtered socks
[...]

Aunque lo mismo interesa buscar algunos servidores con NT:
nmap -sS -O -p 80 **objetivo***/24.*

Más ejemplos,

*nmap -sS -PI -v ****
Starting nmap V. 3.00 (www.insecure.org/nmap)
*Host (***.***.***.***) appears to be down, skipping it.*
Note: Host seems down. If it is really up, but blocking our ping probes, try -P0

...mmm parece ser que con un escaneo (usando ping) normalillo no basta :P

*nmap -sS -PT -v ****
Starting nmap V. 3.00 (www.insecure.org/nmap)
*Host (***.***.***.***) appears to be up ... good.*
*Initiating SYN Stealth Scan against (***.***.***.***)*
Adding open port 80/tcp
Adding open port 53/tcp
Adding open port 22/tcp
Adding open port 443/tcp
Adding open port 25/tcp
The SYN Stealth Scan took 31 seconds to scan 1601 ports.
*Interesting ports on (***.***.***.***):*
(The 1584 ports scanned but not shown below are in state: closed)
Port State Service
22/tcp open ssh

25/tcp open smtp
53/tcp open domain
79/tcp filtered finger
80/tcp open http
137/tcp filtered netbios-ns
138/tcp filtered netbios-dgm
139/tcp filtered netbios-ssn
143/tcp filtered imap2
389/tcp filtered ldap
443/tcp open https
901/tcp filtered samba-swat
3306/tcp filtered mysql
5432/tcp filtered postgres
6346/tcp filtered gnutella
6699/tcp filtered napster
10000/tcp filtered snet-sensor-mgmt

vaya... pues si que estaba despierto mira tu que cosas :P y resulta que tiene algunos puertos filtrados, ok. En cambio tiene otros como el 25 (servidor SMTP), el 53 (name-domain server) y 22/ssh (secure shell) abiertos.

Vamos a investigar eso un poco ;-) por simple curiosidad:

*C:\>telnet ***.*** 25*

*220 correo.***.*** ESMTP*
MAIL FROM yo mismamente!
250 ok
#en fin, no ha hecho falta el saludo xD. Seguimos.
[...]

y si nos conectamos al puerto 22 sacamos esto **SSH-2.0-OpenSSH_2.5.2p2 xD.**

Molaría saber el sistema operaivo: **nmap -sS -PT -PI -O -T 3** (usamos -O para detectar el SO)

Starting nmap V. 3.00 (www.insecure.org/nmap)
[...]
Remote operating system guess: Linux 2.2.12 - 2.2.19
[...]

nmap -sS -O -vv **NOMBRE/IP_víctima/24**

Con este escaneo barremos una red de clase C mediante un escaneo oculto (stealth SYN) y con la intención de averiguar el SO, además hemos especificado el modo very verbose para obtener más detalles acerca de lo que está ocurriendo.

Estos son algunos ejemplos simples de escaneo, ahora ya estás listo para salir "ahí fuera" a explorar de forma efectiva y segura (?) ya iremos viendo como explotar algunos agujeros de seguridad más a fondo en la revista para los cuales es buena idea realizar un análisis previo del sistema del que queremos comprometer la seguridad, no te pierdas los próximos artículos ;).



Para tus primeras...

Para tus primeras pruebas, te recomendamos utilizar como víctima para tus escaneos el servidor de Hack x Crack que está en la IP 80.36.230.235, que para eso está ;)

SERIE RAW: CONOCIENDO PROTOCOLOS Y SU SEGURIDAD. RAW3: IRC (INTERNET RELAY CHAT)

0. INDICE:

1. DOCUMENTACIÓN

1. 1. Un lugar donde vivir

Un mes más, tenéis con vosotros la serie RAW, pero esta vez para presentaros algo más que un protocolo. Y es que el IRC es más que eso, ya que es también un lugar donde vivir. Realmente, hay una diferencia cualitativa entre pasar tus ratos en Internet visitando páginas Web y bajando algún que otro gigabyte, y entre tener además una residencia fija en la red, donde se sabe que se te podrá encontrar siempre que estés en persona en la red (es decir, que no esté sólo tu PC conectado mientras tú te tomas unas cervezas).

Por supuesto, tú no eres el único habitante de esa residencia, si no que la compartes con millones de usuarios de todo el mundo que han escogido alguna red de IRC como su hogar en la red.

¿Y por qué IRC, y no cualquiera de los otros famosos sistemas de mensajería, como Messenger, ICQ, ...? Pues hay muchos motivos, pero yo creo que los más importantes son sobre todo dos:

- En primer lugar, por la libertad, ya que IRC no es un sistema propietario, como lo puede ser Messenger (Microsoft), ICQ (Mirabillis), y muchos otros. Al ser IRC un estándar libre, tienes cientos de redes en todo el mundo para escoger la que más te guste. Hay redes al gusto de cada uno. Por ejemplo, si te gusta la anarquía (en el sentido de que no hay prácticamente bots de servicio) puedes escoger EfNet, en cambio, si te gusta ser controlado, puedes escoger el IRC Hispano.
- En segundo lugar, porque desde 1988 (cuando Jarkko Oikarinen puso en funcionamiento el primer servidor de IRC), las redes de IRC han ido creciendo y convirtiéndose en un complejo universo autosuficiente. Y digo autosuficiente porque puedes conseguir cualquier cosa utilizando tan sólo el IRC. Podría hablaros de los miles de canales de warez, y de otras muchas cosas, pero ese no es el objetivo de este artículo, en el cual no voy a hablaros sobre el mundillo del IRC, si no sobre el protocolo que hace que la cosa funcione.

1.2. Como siempre, empezamos con los RFCs

Aunque muchos de vosotros paséis del tema, ya que para saciar la curiosidad de la mayoría bastará con leer el artículo (que para eso está), aun así es mi deber dar las referencias para los que queráis profundizar más en el tema. Considerando la gran cantidad de documentación que circula sobre IRC, me veo en la necesidad de insistir en que aquí sólo voy a hablar de IRC como protocolo. Así que este no es otro más de los miles de artículos clónicos en los que se explican las mismas barbaridades de siempre (takeovers, nukes, etc, etc). Así que cambiemos el chip, e imaginemos que somos ingenieros de la **IETF** (Internet Engineering Task Force), y no simples usuarios curiosos. ;-)

Con respecto a los RFCs, hay básicamente 4 que abarcan todos los aspectos del protocolo IRC, aunque en este artículo hablaremos un poco también de un quinto documento: el de **IDENT**.

El RFC que empieza explicando los conceptos básicos de la **arquitectura** del IRC es el **RFC 2810** (ftp://ftp.rfc-editor.org/in-notes/rfc2810.txt). Si conocéis ya bien el funcionamiento del IRC os podéis saltar este RFC, aunque nunca está de más mirarlo por si acaso os enteráis de algo nuevo.

Hay otro RFC, que es el que profundiza en el tema de la administración de **canales**. Se explican todos los tipos de canales, así como sus modos. Este es el **RFC 2811** (ftp://ftp.rfc-editor.org/in-notes/rfc2811.txt).

El tercer RFC es el más interesante, y es sobre el que más hablaremos a lo largo del artículo. En él se explica el protocolo utilizado entre un **cliente** y un servidor de IRC, y es el **RFC 2812** (ftp://ftp.rfc-editor.org/in-notes/rfc2812.txt).

Por último, en el **RFC 2813** (ftp://ftp.rfc-editor.org/in-notes/rfc2813.txt) se entra en detalle en el protocolo utilizado para la comunicación de dos o más **servidores** de IRC. Si hemos leído el primer RFC ya sabremos que la arquitectura de IRC es distribuida, pero tranquilos, que ahora mismo os lo explico. :-)

1.3. Arquitectura del IRC

Esto será un breve resumen, ya que escaparía de la intención de este artículo el hacer una descripción detallada de las cuestiones que están más allá de la mera descripción de un protocolo.

Podríamos estructurar, simplificando mucho, la arquitectura del IRC en 3 capas:

- redes
- servidores
- canales

1.3.1. Redes

Existe una gran variedad de redes de IRC totalmente independientes. Cada red sería como un mundo aparte. Por ejemplo, el canal #hackxcrack puede existir tanto en EfNet como en el IRC-Hispano, pero nada tendrán que ver entre sí ambos canales, ya que en cada uno habrá una gente diferente, y unas normas diferentes.

Nombro aquí unas cuantas redes con algún **comentario personal** (basado únicamente en mi experiencia) sobre sus características:

- **EfNet**: la primera red que se mantiene desde el comienzo del IRC hasta nuestros días. Técnicamente, funciona bastante bien (pocos splits, poco lag, etc) y, al ser una red tan antigua, tiene ya muchos canales firmemente consolidados en los que se pueden encontrar gran cantidad de recursos (información, software, etc). Su principal inconveniente podría ser quizá la anarquía, ya que no existen bots de servicio para registro de nicks y canales, pero eso es según se mire, ya que muchos lo pueden considerar una ventaja. Es quizá la red con mayor número de usuarios (en el momento en que escribo esto hay casi **124000 usuarios** conectados)
- **Undernet**: Otra red clásica. Hace años se movía mucho warez por ahí, pero las cosas ya no están como antes. Aún así sigue siendo una red con bastante vida.
- **Dalnet**: Bastante parecida a Undernet. También se movía mucho warez, y cuenta también con bastantes usuarios.
- **IRC-Hispano**: Es una red española, a diferencia de las otras. La mayoría de los usuarios son españoles, y no se ven apenas sudamericanos. Esta red, sobre todo en los últimos años, ha sufrido gran cantidad de polémicas por usuarios descontentos por la política de sus administradores. Yo, personalmente, hace más de un año que juré no volver a pisarla, pero no voy a entrar aquí en polémicas. ;-)
- **Red latina**: Otra red de habla hispana pero, al contrario que la anterior, ésta está poblada sobre todo por sudamericanos, y apenas se ven españoles. Es una red bastante insegura, aunque explicar el por qué se saldría del objetivo de este artículo. 0:)

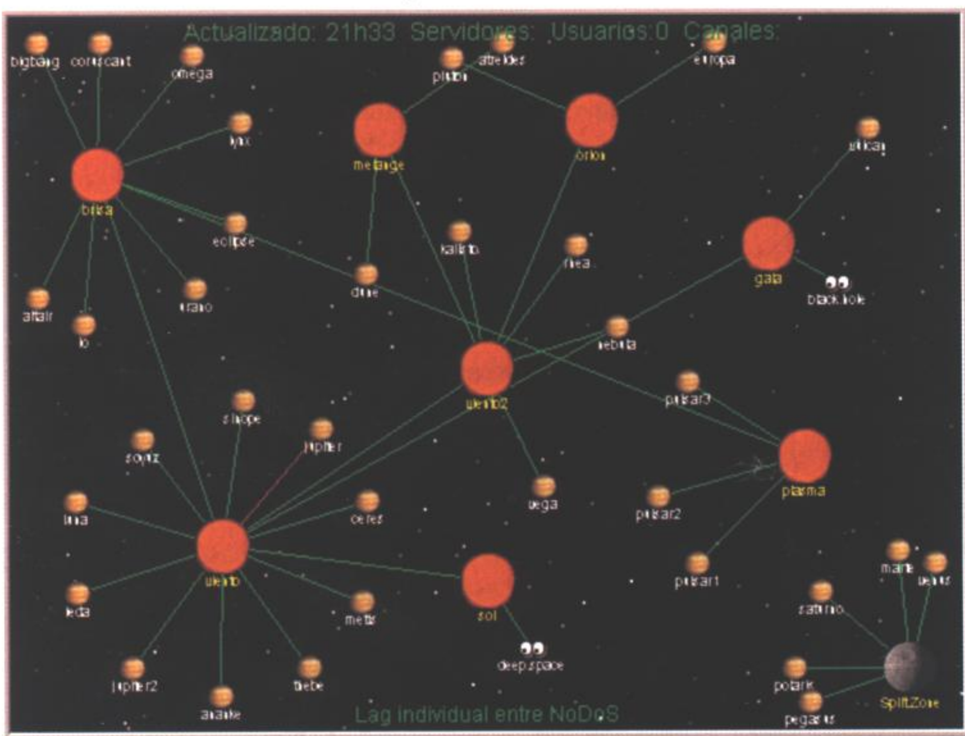
1.3.2. Servidores

Cada red de IRC está formada a su vez por un cierto número de servidores. Todos los servidores forman una red única (o así debería ser) cuya finalidad es hacer que la naturaleza distribuida del servicio de IRC sea transparente al usuario. Es decir, independientemente del servidor al que te conectes, tendrás los mismos servicios y tendrás comunicación con las mismas personas y los mismos canales, siempre y cuando los servidores pertenezcan a la misma red. Esto en la práctica no es tan perfecto como lo pinto aquí, ya que hay muchos problemas como los splits, o las diferencias en los detalles de configuración entre servidores.

La arquitectura distribuida de las redes de IRC permite tener un mayor número de usuarios, aunque el sistema es poco eficiente, y cada

vez que se conecta un nuevo usuario aumenta la carga de todos los servidores (y no sólo del servidor al que se ha conectado), ya que todos los servidores tienen que mantener unas tablas de datos comunes, así como una comunicación constante de todo tipo de eventos, para "garantizar" la transparencia de la arquitectura de la red de cara al usuario.

Por poner un ejemplo, en el instante en que escribo esto, hay **4081 usuarios** conectados al servidor de EfNet al que estoy yo conectado, mientras que en total, en todo EfNet, hay casi 124000 usuarios.



Mapa de servidores del IRC-Hispano. En http://wjr.org/?target=ircmaps&state=expand tenéis el mapa de servidores de EfNet.

A modo de ejemplo, estos son algunos servidores de algunas redes. Se pueden encontrar listas completas en la Web de cada red, o bien en la ventana de conexión de cualquier cliente de IRC:

Efnet:	irc.isdnet.fr
	irc.homelien.no
	efnet.demon.co.uk
Dalnet:	irc.dal.net
	irc.eu.dal.net
Undernet:	eu.undernet.org
	us.undernet.org

1.3.3. Canales

Dentro de cada red, habrá un cierto número de canales de todo tipo. Hay canales temáticos (distintos estilos o grupos de música, distintos hobbies, etc), así como canales de colectivos de toda índole (desde grupos de warez, hasta simples grupos de amigos). Cualquiera puede crear un canal con tan sólo escribir un comando, por lo que el número de canales es realmente grande. Por ejemplo, mientras escribo esto, en EfNet están funcionando en este instante **44345 canales**.

dicho, la arquitectura distribuida de una red de IRC es transparente al usuario, por lo que cualquier usuario conectado a una determinada red debería ver los mismos canales que cualquier otro usuario que esté conectado a otro servidor de la misma red. Sólo hay una salvedad a esto, y es que existe la posibilidad de crear canales locales para cada servidor, y son los canales que, en lugar de comenzar por **#** comienzan por **&**.

Por ejemplo, si estoy en el servidor irc.isdnet.fr de EfNet y creo el canal **#hackxcrack**, cualquier usuario de EfNet podrá entrar en ese canal, esté en el servidor que esté. En cambio, si creo el canal **&hackxcrack**, sólo podrán entrar en él los usuarios de EfNet que además estén conectados a través del servidor

irc.isdnet.fr.

Tipos de usuarios de un canal :

Para comprender muchas de las cosas que explicaré a lo largo del artículo, hay que conocer los 3 tipos básicos de usuarios en cualquier canal:

- **operadores (@)** : son los usuarios con privilegios para administrar la configuración del canal. En la lista de usuarios del canal, su nombre aparecerá precedido por una **arroba @**. A lo largo del artículo veremos en qué consisten exactamente los privilegios de un operador. Es importante que no confundamos un **operador de canal** con un operador de la red o del servidor (a veces llamados **IRCops**).
- **voz (+)** : su nombre aparece precedido por una **cruz +**. En los **canales moderados** (ya veremos más adelante lo que significa esto), son los únicos que pueden hablar en el canal sin ser operadores. En canales no moderados, se utiliza muchas veces como signo de **distinción**, aunque no les da ningún privilegio sobre la administración del canal.
- **usuarios llanos** : todos los que no tengan ni @ ni **+**. :-)

1.4. Seguridad en IRC

¿Os creéis que os voy a contar aquí lo que podéis encontrar en cualquiera de los millones de tutoriales sobre "hacking" en IRC? Mi objetivo es aportar algo nuevo, así que si queréis conocer los típicos truquitos de toda la vida, recurrid al omnisciente **Google**. :-)

En este artículo me voy a centrar sólo en el protocolo IRC.

Lo que sí que puedo aportar con respecto a seguridad, y que no encontraréis en cualquiera de esos tutoriales clónicos, es la seguridad en el protocolo **DCC** (utilizado para conexiones punto a punto dentro de IRC, por ejemplo para chats privados, o para envío de archivos), pero el DCC es un protocolo en si mismo, y merece un artículo entero para el solito, así que paciencia, y esperad al próximo mes para conocer los entresijos del DCC. ;-)

2. PROTOCOLO IRC

Vamos ya al grano.

Podríamos dividir el IRC como protocolo en 3 capas de abstracción:

- script
- aplicación cliente
- protocolo raw

La capa **script** sería aquella en la que se realizan las acciones mediante menús e iconos. Cuando se pulsa uno de estos iconos o menús, lo que hace nuestro script en realidad es ejecutar una serie de comandos de las capas inferiores que veremos ahora mismo.

La capa **aplicación cliente** es quizá la más conocida por parte de los asiduos al IRC. Son aquellos comandos que escribimos directamente desde el cliente de IRC, utilizando como prefijo la famosa barra /.

Al contrario de los que algunos piensan, estos comandos no son los que envía a pelo nuestro cliente de IRC al servidor. Al igual que los menús y los iconos, siguen siendo simplificaciones para hacer más fácil el manejo al usuario.

En la capa de **protocolo raw** es en la que se envían a pelo los comandos que entiende el servidor, y ésta es en la que se va a centrar el artículo. Éste es el protocolo que tendréis que utilizar si conectáis por **Telnet** a un servidor de IRC, ya que es el único que entiende el servidor. Algunos comandos difieren muy poco entre la capa raw y la anterior (en muchos casos, la única diferencia es que en la capa raw el comando es el mismo pero sin la barra / delante), pero otros son bastante más complejos (por ejemplo, los comandos de DCC, que explicaré en el próximo artículo).

Nos olvidamos de las demás capas, y nos centramos a partir de ahora tan sólo en la capa **raw**.

2.1. Comandos de conexión

2.1.1. Conectando con el servidor

¿Tenéis ya vuestro cliente de **Telnet** calentando motores? Espero que así sea, porque entramos ya directamente con el tema.
El puerto estándar para los servidores de IRC es el **6667**, aunque dependiendo de cada servidor, el servicio podrá encontrarse en otro puerto. Por eso es necesario tener siempre a mano los detalles de cada servidor.
A lo largo del artículo vamos a utilizar el servidor **irc.isdnet.fr** de **EfNet** (EfNet es una red libre, por lo que al utilizarlo como ejemplo no entro en polémicas de partidismos; Además, en EfNet se encuentra el canal **#hackxcrack** en el que podréis encontrarme normalmente), así que esto es lo que tendríais que escribir para lanzar la aplicación de Telnet:
telnet irc.isdnet.fr 6667

2.1.2. Identificándonos

Una vez conectados, tenemos que ser rápidos, ya que el servidor tiene un **timeout**, así que tendremos que tener preparados los comandos de identificación para lanzarlos rápidamente. Son 2 los comandos a ejecutar para completar la entrada en el servidor:

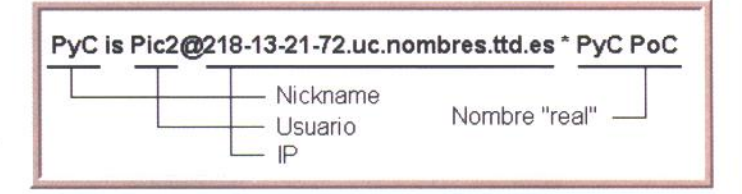
2.1.2.1. Comando NICK

El primer paso es muy sencillo, simplemente tenemos que indicar nuestro nickname. Si, por ejemplo, queremos que nuestro nick sea PyC, sólo tenemos que escribir:
NICK PyC
Si nos dice que el nick ya está en uso, podemos volver a ejecutar el comando **NICK** tantas veces como queramos, hasta que encontremos un nick libre.
Después de haber completado el proceso de identificación (con este comando y el que explico a continuación), podremos utilizar el comando **NICK** cada vez que queramos cambiar nuestro nickname.

2.1.2.2 Comando USER

El segundo paso consiste en dar el resto de datos para identificarnos, es decir: el nombre de usuario, nuestro nombre "real", y los modos de usuario.

Si sois ya usuarios de IRC, supongo que miles de veces habréis hecho un WHOIS a alguien, y la primera línea que os habrá respondido el servidor será algo parecido a esto:



Como vemos aquí, lo primero que nos muestra el whois es el **nick** (PyC), lo segundo el nombre de **usuario** (Pic2), lo tercero la **IP**, o un DNS asociado a esa IP (218-13-21-72.uc.nombres.ttd.es), y por último el **nombre "real"** del usuario (PyC PoC). Tanto en el campo de nombre de usuario, como en el de nombre real en principio podemos poner lo que queramos, por lo que no le demos más vueltas.
El único detalle que hemos de saber es que en el campo de nombre real es el único en el que se pueden separar palabras por espacios.
Lo único importante a la hora de identificarnos sería el campo de **modos**, pero en realidad en muchos servidores no está implementado el cambio de modos durante la identificación (por ejemplo, en el que estamos utilizando, irc.isdnet.fr, no está implementado), por lo que normalmente hay que cambiar los modos después de habernos identificado.
El campo de modos en la identificación permite poner automáticamente los modos de usuario **+w** y **+i**, mediante una máscara de bits (**bit 2** para **+w**, y **bit 3** para **+i**). Por tanto, estos

son los valores que habría que poner en este campo según los modos que queramos activar:

Modos activos	máscara para la identificación
Ninguno	0
+w	4
+i	8
+w y +i	12

Aunque insisto en que no es mi objetivo explicar en este artículo el funcionamiento del IRC a nivel de usuario, os recuerdo que:

- el **modo +i** es el modo de usuario **invisible**, que permite que tu nick no aparezca en la lista de usuarios de un canal cuando se hace un **/who** de ese canal. Por lo demás, el usuario es perfectamente visible mediante /whois, o para los usuarios que comparten algún canal con él, por lo que desde luego esta invisibilidad no es comparable a la del anillo único. ;-)
- el **modo +w** sirve para recibir los mensajes **wallops**, que son los mensajes para administración de la red que se envían a los operadores y a los demás interesados. En muchas redes este modo no tiene ningún efecto.

Como en nuestro ejemplo el servidor no soporta el cambio de modos en la identificación, nosotros pondremos un 0. Vamos a ver entonces cómo quedaría el comando USER para el usuario PyC:

USER Pic2 0 * : PyC PoC

Veamos campo por campo:

En primer lugar, tenemos el **nombre de usuario**: Pic2.

En segundo lugar, la **máscara de modos**: 0.

En tercer lugar, un campo que **no se utiliza**, por lo que podemos poner cualquier cosa: *.

Por último, e incluyendo espacios si queremos, separado del resto de campos por dos puntos (:), tenemos el **nombre "real"**: PyC PoC.

¿De dónde salen entonces el resto de campos del whois? Es decir: el **nickname**, y la **IP**, o DNS. Pues el **nickname** lo especificamos precisamente en el primer comando (comando **NICK**), y la **IP** la obtiene directamente el servidor de la conexión que tenemos establecida (cualquier conexión TCP/IP es una conexión punto a punto que requiere de forma imprescindible que los dos interlocutores conozcan la dirección IP del otro).

2.1.2.3. ¿Y si fuésemos operadores?

Si fuésemos operadores (IRCops, o como los quieras llamar) de la red, tendríamos que utilizar un comando adicional para identificarnos como operadores. Para eso se utiliza el comando **OPER**, con la siguiente sintaxis:

OPER nombre password

El primer campo es el nombre del operador, y el segundo su password de operador.

2.1.2.4 El IDENT

Seguramente habréis oído hablar más de una vez acerca del famoso **IDENT**, pero es probable que no tengáis más que una idea vaga de lo que es. Probablemente lo habréis visto en la configuración de vuestro cliente de IRC, y por eso seguramente habéis pensado que forma parte del protocolo IRC, pero no es así. El protocolo **IDENT** es un protocolo totalmente independiente y, de hecho, no sólo se usa en relación con IRC. Este protocolo está asignado al **puerto 113 de TCP**. En el caso concreto que nos ocupa no podemos estudiarlo por **Telnet**, ya que **nosotros seremos los servidores de ident**, y no los clientes. Algunos servidores de IRC no permiten entrar a ningún usuario que no tenga un servidor de ident. Durante un tiempo, el servidor que utilizamos para el ejemplo restringía sus conexiones de esta forma, así que espero que para cuando salga a la luz el artículo no tengamos este problema. Este problema, al menos en los casos en que yo lo he visto, suele ser siempre cuestión de un servidor concreto,

y no de toda la red, por lo que si un servidor no nos permite entrar sin tener servidor **identd** activo, podemos buscar otros servidores de la misma red hasta encontrar uno que sí que nos lo permita.

El protocolo **IDENT** viene especificado en el **RFC 1413** (ftp://ftp.rfc-editor.org/in-notes/rfc1413.txt).

Como resumen, éste protocolo se utiliza para verificar la **identidad del usuario de una conexión TCP/IP** específica, que viene identificada por un **par de puertos** (el puerto del cliente, y el puerto del servidor).

2.1.3. Más sobre los modos de usuario (comando MODE)

Tanto si nuestro servidor no permitía cambiar los modos de usuario directamente en la identificación, como si queremos fijar otros modos de usuario aparte del +w y el +i, podemos utilizar en cualquier momento el comando **MODE**.

Este es un resumen de los modos de usuario que merecen mención:

- **modo +i**: ya hablé de él en el punto anterior. :P
- **modo +w**: ídem. :P
- **modos +o y +O**: indica que el usuario es un operador de la red, o un operador local, respectivamente. Si no te has identificado previamente con OPER, olvídate de este modo. ;-)
- **modo +s**: este modo, si está implementado en el servidor, puede ser bastante interesante. En el servidor del ejemplo (irc.isdnet.fr), este modo no está implementado. Podemos probarlo, por ejemplo, en el IRC-Hispano. Este modo sirve para recibir mensajes del servidor, referentes a g-lines, kills, etc. ¿Y qué utilidad puede tener esto? Pues se puede extraer información útil si miramos con detenimiento los mensajes. Permanezcamos un rato conectados logeando los mensajes del servidor y veremos, por ejemplo, algo como esto:

[16:13] -luna.irc-hispano.org- * Notice -- sol.irc-hispano.org adding GLINE for *@usuario69.lco.es, expiring at 1049296487 (Wed Apr 2 17:14:47 2003): Proxy abierto o Sub7 server, desinfectese**

El servidor nos acaba de decir que en la dirección usuario69.lco.es está funcionando, o bien un servidor sub7, o bien un Proxy. Basta con que ahora hagamos un escaneo de los puertos de esa IP para encontrar su vulnerabilidad (en caso de que sea un sub7) o su "servicio" disponible (en caso de que sea un Proxy). Son muy variopintos estos mensajes, pero nos pueden dar en muchos casos información útil.

Otra utilidad más rebuscada de esto, concretamente en el IRC-Hispano, es conseguir la IP de un usuario mediante un poco de ingeniería social. Lo cuento sólo como anécdota, ya que tiene poco interés, pero al menos es curioso. Fue uno de mis muchos experimentos chorras. El IRC-Hispano utiliza un sistema de direcciones virtuales que oculta la IP de los usuarios, por lo que hay que recurrir a diversas tretas (que no voy a detallar porque se sale del tema) para conseguir estas IPs. Como sabréis los residentes en el IRC-Hispano, existe un bot de servicio llamado **_antispam** que se encarga de que los únicos que se lucren en el IRC-Hispano sean sus administradores, intentando evitar que nadie publique URLs de páginas Web con supuestos fines publicitarios, a no ser que paguen previamente. El que mencione una de estas URLs en un canal en el que resida _antispam, será **g-lineado** (expulsado de la red) automáticamente. Así que lo que hice en esta ocasión, fue conseguir mediante un mínimo de esfuerzo de ingeniería social (algo en plan: "¿alguien sabe una Web donde haya tal o cual?") que el usuario en cuestión escribiese una de las URLs malditas. Inmediatamente después de escribirla, fue g-lineado por el bot _antispam. Inmediatamente miré los mensajes del servidor (que podía ver gracias al modo +s), y vi algo parecido a esto:

[16:58] -luna.irc-hispano.org- * Notice -- sol.irc-hispano.org adding GLINE for**

***@61-136-23-69.dialup.uni2.es, expiring at 1048696764 (Wed Mar 26 17:39:24 2003): Publicidad no autorizada. Si desea contratar servicios publicitarios, consulte http://www.irc-hispano.org/servicios/**

Ese g-line correspondía con el usuario que acababa de ser expulsado por _antispam, por lo que ya tenía su IP: 61.136.23.69.

- **modo +x**: Claro que, si somos unos maestros de la ingeniería social, podemos conseguir que el usuario del IRC-Hispano cuya IP queremos conocer escriba en su cliente de IRC: **/mode** usuario **-x**. Donde usuario es su nick. Con el modo –x inhabilitamos la dirección virtual de ese usuario, por lo que cuando le hagamos un **whois** veremos directamente su IP, y no una dirección virtual.

Como hemos visto al hablar de este último modo, un usuario sólo puede cambiar sus propios modos (es de cajón, ¿no?). La forma de hacer esto mediante el protocolo RAW es prácticamente igual que desde la aplicación cliente. Por ejemplo, para activar el modo +s, siendo el usuario PyC, haremos:

MODE PyC +s

2.1.4. Comando QUIT (y el /quote)

Ya que estoy siguiendo más o menos el orden en el que se explican los comandos en el RFC (al menos los que considero importantes), seguiré con el comando **QUIT**. No creáis que es demasiado pronto para cerrar la conexión, porque quizá os convenga saber que para probar el resto de comandos podéis cerrar ya vuestra aplicación de Telnet. Y es que cualquier cliente de IRC proporciona un mecanismo para enviar mensajes raw al servidor, y es el comando **/quote**. Después de un /quote, todo lo que se escriba será enviado tal cual al servidor, por lo que deberá seguir el protocolo raw de IRC (que es el protocolo que explico en este artículo).

Empezamos probando el comando QUIT desde nuestro cliente de **Telnet**:

QUIT :Hasta pronto!

Con esto cerramos la conexión con el servidor de IRC, enviando como mensaje de despedida "Hasta pronto!". Este mensaje lo leerán los usuarios de todos los canales en los que estés cuando cierres.

Si queremos enviar esto mismo desde nuestra **aplicación cliente de IRC** (mIRC, kvirc, xchat, o el que sea), sólo tendremos que escribir:

/quote QUIT :Hasta pronto!

Y esto será igual para todos los comandos explicados en este artículo, incluidos los ya explicados, excepto el comando **USER**.

2.2. Comandos de canales

2.2.1. Entrando en canales (comando JOIN)

Si estamos acostumbrados a manejar aplicaciones cliente de IRC, veremos que de momento hay pocas diferencias entre los típicos comandos con la barra / y el protocolo raw de IRC, y eso sigue siendo igual para el comando para entrar en un canal:

JOIN #hackxcrack

Es el comando raw para entrar en el canal **#hackxcrack**.

Este canal existe en **EfNet** (que es precisamente donde estamos haciendo las pruebas), y es probable que me encontréis ahí, así que podéis pasaros para vuestras pruebas. ;-)

Como particularidades del comando **JOIN**, puedo decir 4 cosas:

En primer lugar, algunos canales necesitan un **password** para entrar. Supongamos que el password del canal **#hackxcrack** es pcpasoapaso. En ese caso el comando para entrar en el canal será:

JOIN #hackxcrack pcpasoapaso

La segunda particularidad es que puedes entrar en varios canales utilizando sólo un comando **JOIN**, separando los canales por comas. Por ejemplo:

JOIN #hackxcrack,#spanishwarez

Tercera curiosidad, si escribimos:

JOIN 0

Saldremos simultáneamente de todos los canales en los que previamente hayamos hecho un **JOIN**.

Por último, si hacéis **JOIN** a un canal que no ha sido creado por nadie previamente, automáticamente ese canal será creado y tú serás operador del canal (la famosa @). En el momento en que abandone el canal el último usuario, el canal desaparecerá (es decir, simplemente será borrado de las tablas de datos que mantienen en tiempo real los servidores).

2.2.2. Saliendo de canales (comando PART)

Una vez más, el comando nos será ya familiar:

PART #hackxcrack

Sirve para salir del canal **#hackxcrack**. Si queremos incluir un mensaje de despedida, podemos hacer:

PART #hackxcrack :me piro!

En el comando PART también podemos poner varios canales, separados por comas.

2.2.3. Modos de canal (comando MODE de nuevo)

El comando **MODE** no sólo sirve para cambiar los modos de usuario, sino también los modos de canal. En el **RFC 2811** tenéis detallados todos los modos de canal, así que sólo os resumo los más importantes:

- **modo +i (invite only)** : sólo pueden entrar en el canal los usuarios que son invitados explícitamente, mediante el comando **INVITE**, o bien aquellos que están en la **lista de excepción** de invite del canal (están permanentemente invitados).
- **modo +m (moderated)** : sólo pueden hablar en el canal los usuarios que son **operadores (@)**, y los que tienen **voz** (+).
- **modo +n (no outside messages) :** sólo pueden hablar en el canal los usuarios que previamente han entrado en el mismo. Aunque esto os suene raro, si un canal está en modo –n (es decir, no está en modo +n) cualquiera puede hablar sin necesidad de estar dentro del canal. Ya veremos más adelante cómo se hace esto.
- **modo +s (secret)** : el canal no aparecerá en la lista de canales de un usuario cuando se haga **whois** a este usuario.
- **modo +t (topic)** : el topic del canal sólo puede ser cambiado por los operadores (@).
- **modo +k (key)** : sólo se puede entrar al canal si se conoce el password.
- **modo +l (limit)** : limita el número máximo de usuarios que pueden entrar en el canal.

Todos estos modos sólo pueden ser cambiados por los operadores (@) del canal.

Como ejemplo, para poner el canal **#hackxcrack** en modo secreto (suponiendo que seamos operadores del canal) haremos:

MODE #hackxcrack +s

Si luego nos arrepentimos, y queremos quitar este modo, podemos hacer:

MODE #hackxcrack –s

Para limitar el número de usuarios a 22:

MODE #hackxcrack +l 22

A partir de este momento, cuando intente entrar (con **JOIN**) alguien en el canal, habiendo ya 22 personas, no podrá hacerlo hasta que alguien salga del canal.

Como último ejemplo, para poner el **password** pcpasoapaso en el canal **#hackxcrack**, haremos:

MODE #hackxcrack +k pcpasoapaso

A partir de este momento, para entrar en el canal habrá que hacerlo con el comando:

JOIN #hackxcrack pcpasoapaso

Para lo cual, por supuesto, habrá que conocer previamente el password pcpasoapaso. ;-)

También se pueden poner simultáneamente varios modos, por ejemplo:

MODE #hackxcrack +ntk pcpasoapaso

Activaría simultáneamente los modos **+n**, **+t**, y **+k** con password pcpasoapaso.

2.2.4. Modos de usuario para un canal (comando MODE una vez más)

No termina aquí la utilidad del comando **MODE**. Pero no penséis que éste es el comando más polifacético, ya que el comando **PRIVMSG**, que veremos más adelante, le supera con creces. :-)
Los modos de usuario para un canal son aquellos que aplican unas normas o unos privilegios para un determinado usuario sólo dentro de un determinado canal. Todos ellos pueden ser fijados sólo por un operador del canal. Estos modos vienen también especificados en el **RFC 2811**, así que resumo aquí los más importantes:

- **modo +o (operator)** : convierte a un usuario en **operador** del canal (la famosa @).
- **modo +v (voice)** : da **voz** a un usuario en un canal (el famoso +).
- **modo +b (ban)** : sirve para crear máscaras que **denieguen el acceso** al canal a un usuario, o bien a un grupo de usuarios, según cómo se forme la máscara. Esto lo explicaré más adelante.
- **modo +e (exception)** : sirve para **permitir acceso** a un usuario concreto que esté dentro de las máscaras de denegación. Ya lo veremos luego, que suena un poco lioso. :-)
- **modo +I (invitation mask)** : permite crear máscaras para que determinados usuarios puedan entrar en un canal en modo **invite only (+i)** sin necesidad de que nadie les invite explícitamente.

El ejemplo más sencillo es el de los modos **+o** y **+v**. Por ejemplo:
MODE #hackxcrack +o PyC
Da @ al usuario PyC.

El tema de las **máscaras** (los otros 3 modos) es ya más complejo. Vamos a ir viendo distintos tipos de máscaras mediante ejemplos con los distintos modos.

En primer lugar, imaginemos que queremos que en el canal #españa sólo entre gente desde una IP española. Para ello podemos hacer:
MODE #españa +i
MODE #españa +I *!*@*.es
El primer comando pondrá el canal en modo **invite only**, por lo que sólo podrán entrar los usuarios que sean invitados explícitamente, o bien los que estén en la lista de excepción de invitaciones. Esta lista es la que se administra con el modo **+I**, y eso es precisamente lo que hacemos con el segundo comando, en el que especificamos que todos aquellos usuarios, se llamen como se llamen, que vengan de un dominio **.es**, puedan entrar al canal sin necesidad de ser invitados explícitamente.

Otra forma de hacer esto mismo, sería la siguiente:
MODE #hackxcrack +b *!*@*
MODE #hackxcrack +e *!*@*.es
Con el primer comando, **denegamos el acceso a todos los usuarios**. Con el segundo, hacemos una **excepción** a esa lista de denegaciones de acceso, permitiendo que sí que puedan entrar los que vengan desde un dominio **.es**.

Ahora vamos a ver cómo denegar el acceso al usuario cuyo **nickname** sea PyC:
MODE #hackxcrack +b PyC!*@*
Si PyC tiene 2 dedos de frente, bastará con que se cambie el nick para poder entrar (con el comando **NICK**).

A veces hay canales en los que, por el motivo que sea, no quieren que los usuarios utilicen determinados **scripts**. Los scripts suelen poner por defecto un determinado **nombre de usuario** a la hora de identificarse, por lo que es fácil identificarlos siempre y cuando el usuario

no haya modificado este dato a posteriori. Supongamos, por ejemplo, que queremos **banear** (denegar el acceso) a todos los usuarios que utilicen el script **Ircap 7.1**. Si vemos el **whois** de un usuario de Ircap 7.1, será parecido a éste:

PyC is ~ircap71@201.Red-22-15-15.pooles.rima-tde.net * http://www.ircap.net !

Como vemos, el campo **nombre de usuario** es ~ircap71, que es el que pone por defecto este script. Por tanto, podemos hacer:

MODE #hackxcrack +b *!*ircap71@*

Para denegar el acceso a todos los usuarios que tengan **ircap71** en el nombre de usuario.

Por último, podemos **banear** una **IP** específica de la siguiente forma:

MODE #hackxcrack +b *!*@201.Red-22-15-15.pooles.rima-tde.net

Está claro, ¿no? ;-)

2.2.5. El Topic del canal (comando TOPIC)

El **topic** es, por así decirlo, un **texto de presentación de un canal**. Cuando entras a un canal, lo primero que recibes es el texto del topic.

Las utilidades del topic son muchas: dar una serie de **normas** básicas para los usuarios del canal (estos topics suelen permanecer fijos), dar alguna URL interesante de **actualidad** (estos topics suelen cambiar de un día para otro), o simplemente escribir la **chorrada** de turno (he llegado a ver conversaciones enteras entre 2 o más operadores cambiando el topic en lugar de escribiendo en el canal).

La primera utilidad del comando **TOPIC** es simplemente ver el texto actual del topic:

TOPIC #hackxcrack

Nos mostrará el topic actual del canal #hackxcrack.

Para modificar el topic, simplemente añadiremos un campo más:

TOPIC #hackxcrack :Bienvenidos al canal de PC Paso a Paso.

El topic sólo puede ser modificado por los **operadores** del canal, a no ser que el canal esté en **modo –t** (es decir, que no tenga activo el modo +t).

2.2.6. Buscando canales (comando LIST)

Aquí la cosa se pone fea. :-(

¿Habéis usado alguna vez la función **LIST CHANNELS** de mIRC para buscar canales que contengan una determinada cadena de texto?

mIRC List Channels
List channels which match these criteria:
Using file: EFnet.txt
Get List!
Match text: mp3
Apply
Match text in topics
OK
Number of people on a channel:
min: 0 max: 9999
Lock
Hide channels which match:
Hide non-alphanumeric channels

Menú en mIRC para buscar la cadena "mp3" en la lista de canales de EfNet.

Lo que hace esta función de mIRC en realidad es pedir al servidor la lista de todos los canales, y luego el propio mIRC hace la búsqueda de la cadena de texto dentro de la lista que le ha proporcionado el servidor. Por tanto, no hay forma de que podáis hacer una búsqueda selectiva lanzando comandos raw desde Telnet. :-(

Para obtener la **lista completa de canales**, con sus topics, y el número de usuarios, basta

con que ejecutéis:
LIST

2.2.7. Comando INVITE

Por seguir con el orden del RFC, tenemos un comando muy simple, que es para **invitar** a un usuario a un canal, normalmente porque el canal se encuentra en modo **invite only (+i)**:
INVITE PyC #hackxcrack
Con esto invitamos al usuario PyC al canal #hackxcrack.

2.2.8. El comando prohibido (comando KICK)

Normalmente, si eres una persona madura e inteligente, no necesitarás utilizar este comando salvo en situaciones muy concretas. Este comando permite a los operadores **expulsar** a un usuario de un canal. Por ejemplo:
KICK #hackxcrack PyC :nos estas floodeando
Expulsaría al usuario PyC del canal #hackxcrack, exponiendo el motivo: "nos estas floodeando". Si queremos **expulsar permanentemente** a ese usuario del canal, tendremos que **banearle** primero, es decir, aplicar una máscara para denegar su acceso al canal:
MODE #hackxcrack +b PyC!*@*
KICK #hackxcrack PyC :vete y no vuelvas
Claro que, ya he dicho que este ban es poco efectivo, ya que basta con que el usuario se cambie de nick para poder entrar. Según cada caso será más efectivo un tipo de ban u otro. Por ejemplo, para usuarios con **IP dinámica** habría que banear un **subdominio**, siempre y cuando no haya más usuarios legítimos en ese subdominio (en ese caso tendríamos que utilizar además una mascara de excepción con el **modo +e** para los usuarios legítimos). Otra solución para la IP dinámica, es banear según el campo **nombre de usuario**, pero ese campo es fácilmente modificable, por lo que si el tío es un poco listo podría volver a entrar sin problemas.
En el caso de que el usuario tenga **IP fija**, no hay más misterio, baneas directamente la **IP** y se acabó (a no ser que utilice un bouncer, pero eso ya es otra historia).

2.3. Al fin vamos a hablar (el versátil PRIVMSG)

¡Tanto rollo y todavía no sabemos como decir "hola" en un canal! Aquí es cuando vemos al fin una diferencia importante entre los comandos famosos de la barra /, y el protocolo raw. Pensad en todas las cosas de las que todavía no he hablado: escribir en un canal, escribir en una query (mensajes privados entre 2 usuarios), lanzar comandos CTCP, lanzar comandos DCC... ¡Pues todas ellas se hacen con un único comando! Y éste comando es el mágico **PRIVMSG**. Vamos a ver todo esto con detalle.

2.3.1. Mensajes privados (queries)

Para hablar en privado con otro usuario, basta con que hagamos:
PRIVMSG PyC :hola!
El usuario PyC recibirá un mensaje privado diciendo "hola!". Quizá os preguntaréis: ¿cómo que un mensaje privado? ¿y que hay del query de toda la vida?
Y es que supongo que muchos de vosotros tendréis la equivocada idea de que una query es una especie de conexión establecida entre 2 usuarios, pero eso no es así. Esa es la ilusión que os da vuestro cliente de IRC al abrir una **ventana** específica para la query, pero en realidad las queries no son más que sucesiones de mensajes entre 2 usuarios, sin ningún tipo de **establecimiento de conexión**.

2.3.2. Escribiendo en el canal

Para escribir a todos los usuarios de un canal, el formato es el mismo:
PRIVMSG #hackxcrack :hola a todos!
Con esto escribimos en el canal #hackxcrack

la frase "hola a todos!".
Ahora que la cosa es un poco diferente a lo que estáis acostumbrados a hacer con el cliente de IRC, es buen momento para probar el comando **/quote** para hacer esto mismo, desde vuestra aplicación cliente de IRC:
/quote privmsg #hackxcrack :hola a todos!

2.3.3. Escribiendo en un canal en el que no estamos

Escribir en un canal en el que no estemos (no hayamos hecho previamente un **JOIN**) es igual de sencillo, salvo que tiene un prerequisito importante, y es que el canal tiene que estar en **modo –n**, es decir, que no esté en modo +n.
Si tenemos la suerte de encontrar un canal en modo –n podemos hacer una chorrada para pasar el rato. En muchos clientes de IRC, la letra **i mayuscula** se ve prácticamente igual que la letra **ele minúscula**. Si, por ejemplo, tenemos en el canal #lameretes un usuario cuyo nick sea zer0c00l (nick de lamer de primera), podemos ponernos nosotros un nick bastante parecido, que sería zer0c00I. Este nick, cambiando la ele por una i mayúscula, se verá bastante parecido en un cliente de IRC, por lo que será dificil notar la diferencia a simple vista, a no ser que estén los 2 nicks uno al lado del otro. Una vez puesto ese nick, sin entrar en el canal #lameretes, escribiremos:
PRIVMSG #lameretes :pero mira que soy lamer
La gente del canal #lameretes, a no ser que sean un poco avispados, creerá que es zer0c00l quien está diciendo eso, ya que no verán a ningún otro usuario con un nick similar en el canal. Por otra parte, zer0c00l probablemente se volverá loco creyendo que le han hackeado, poseído, o algo así. Por supuesto, esto es una tremenda gilipollez, y cualquiera con 2 dedos de frente lo descubriría (sobre todo porque cualquiera con 2 dedos de frente normalmente tendría el canal en modo +n), pero siempre está bien conocer estas chorradas, ¿no? :-)

2.3.4. El protocolo CTCP

El comando **PRIVMSG** es tan versátil que encapsula él solito todo un protocolo, que es el **CTCP**, o **Client To Client Protocol**. No voy a entrar en muchos detalles sobre el protocolo CTCP, así que sólo hablaré de los comandos más básicos.

2.3.4.1. CTCP PING

El comando más típico de **CTCP** es el **PING**. Su utilidad es doble: en primer lugar, saber si un usuario aparentemente conectado realmente lo está, ya que muchas veces cuando un usuario pierde la conexión con el servidor, durante un tiempo su nick sigue apareciendo como si siguiese conectado (¿os suena el famoso **ping timeout**?), y en segundo lugar mide el **lag** que tienes con respecto a un usuario. ¿Qué es el **lag**? Es simplemente el **retardo** que hay entre 2 usuarios, es decir, el tiempo que transcurre entre que uno escribe algo y el otro lo lee. El lag varía según la carga del servidor o incluso según el ancho de banda disponible de los usuarios. La peor situación es cuando hay varios usuarios hablando, cada uno con un lag diferente pero muy alto, por lo que se forman conversaciones sin sentido en el que cada uno da una respuesta que llega a los demás en un momento diferente.
Yo personalmente utilizo el **PING** de vez en cuando para medir mi propio lag. Lo que hago es lanzarme un **PING** a mi mismo. En el mejor de los casos debería haber un lag de 0 segundos, pero muchas veces este lag puede ser de varios segundos, y con eso ya sabes que, como mínimo, todo lo que leas o escribas tendrá un retraso de tantos segundos.

Para cualquier comando CTCP tenemos que utilizar el **ascii 1**, que aquí representaremos como: □. Como espero que ya sepáis, podeis poner escribir cualquier ascii del cual conozcáis

su valor numérico pulsando la tecla Alt y, sin soltarla, escribiendo su valor en el teclado numérico. Por tanto, para escribir el ascii 1, pulsaremos la Alt+1. Si la aplicación que estás utilizando no te permite escribir este ascii, tendrás que escribirlo en alguna otra aplicación y copiar el ascii desde ahí mediante el clásico copy/paste.

Volviendo al tema, ¿en qué consiste el ping? Pues simplemente es un mensaje corto que contiene tan sólo un **número aleatorio**. En cuanto el receptor lo reciba, tiene que enviar una respuesta que contenga exactamente el mismo número. Nuestra aplicación cronometrará el tiempo transcurrido desde que se envió el número, hasta que éste llegó de vuelta. Para enviar un ping al usuario PyC esto es lo que tenemos que ejecutar:

PRIVMSG PyC :□PING 1048765767□

Para comprender la respuesta que nos devuelven al **PING** hay que hablar de un nuevo comando, que es el **NOTICE**.

Un **NOTICE** tiene exactamente la misma función que un **PRIVMSG**, pero con una salvedad, y es que los **NOTICE** son mensajes que no deben tener respuesta. Es decir, cada vez que una aplicación de IRC recibe un **NOTICE** de cualquier tipo, simplemente lo leerá e interpretará como deba, pero nunca dando ningún tipo de respuesta. Por tanto, la utilidad del **NOTICE** es el enviar mensajes en los que haya riesgo de entrar en bucles infinitos de respuestas.

En nuestro caso, el usuario al que hemos hecho el **PING** nos responderá con el mismo mensaje, pero con la salvedad de que lo hará con **NOTICE** en lugar de con **PRIVMSG**. Si nos respondiese con **PRIVMSG**, nuestro cliente de IRC interpretaría que es a él a quien están lanzando un **PING**, por lo que tendría que responder al mismo, y así se formaría un bucle infinito de pings. Por tanto, la respuesta de PyC a nuestro PING, suponiendo que nuestro nick es fulanin, sería algo como esto:

PyC!~Pic@80-25-32-27.uc.nombres.ttd.es NOTICE Fulanin :□PING 1048765767□

Como vemos, responde con el mismo número, para que podamos confirmar que esta respuesta se corresponde con la petición que habíamos lanzado.

¡¡ATENCION!! Sobre el ping timeout:

Antes de continuar, hago aquí un inciso muy importante, aprovechando que he mencionado antes el **ping timeout**. Los que sepais bien en qué consiste este tipo de caída, probablemente habréis pensado: "este tío no tiene ni idea, el ping timeout no tiene nada que ver con los pings de ctcp". El motivo por el que he mencionado el ping timeout es porque mediante ctcp ping puedes detectar si alguien está caído, pero en realidad con lo que tiene que ver esta caída no es con los pings de ctcp, si no con los **pings del servidor**.

Cuando superas un cierto tiempo de inactividad, el servidor ha de asegurarse de que sigues ahí, para lo cual te lanza un ping bastante parecido a los de CTCP.

Si no respondes a este ping, el servidor cerrará la conexión, asumiendo que se ha quedado colgada. En esto consiste precisamente el ping timeout.

Cuando conectemos por telnet al servidor de IRC, si no hacemos nada durante un tiempo, nos aparecerá un mensaje de PING de servidor, preguntándonos si seguimos ahí. Este mensaje puede contener un número, como en los **CTCP PING**, o bien cualquier otro texto que tenemos que enviar de vuelta. Por ejemplo, en el caso del servidor que estamos utilizando de ejemplo, éste será el PING que nos lanzará:

PING :irc.isdnet.fr

En cuanto veamos esto, tenemos que responder lo antes posible con este comando:

PONG :irc.isdnet.fr

Algunos servidores lanzan el primer PING nada más identificarnos, por lo que hemos de estar

atentos al **mensaje de bienvenida** a la hora de conectar con un servidor, para ver si éste incluye un PING al que tenemos que responder antes de nada.

2.3.4.2. CTCP VERSION

Este mensaje **CTCP** nos puede dar información bastante útil, pero hemos de saber siempre que la respuesta puede haber sido manipulada por el usuario, por lo que podría estar dándonos información falsa.

El objetivo de este mensaje es preguntar a un usuario **qué software** está utilizando como aplicación cliente de IRC.

El formato de la consulta es muy sencillo:
PRIVMSG PyC :□VERSION□

Ante esto, la aplicación cliente del usuario PyC nos responderá de manera automática con una respuesta que identifica el software y el número de versión, así como si tiene algún script funcionando encima, etc.

Respuestas típicas son las siguientes:
mIRC: mIRC v6.01 Khaled Mardam-Bey
Xchat para windows : xchat 2.0.0 Windows 5.0 [i686/449MHz]
mIRC con IRCap 7.1 : mIRC v6.03 Khaled Mardam-Bey • IRcap 7.31 • http://www.ircap.com •
Kvirc 3 : KVIrc 3.0.0-beta2 "T-Rex" : 2003.01.04-12554 : build Sat Jan 4 20:53:34 UTC 2003 : i686-cefikloprstAGT

Como vemos, podemos extraer información útil sobre el sistema operativo del usuario, así como sobre el software que utiliza (que podría tener bugs conocidos, en caso de que quisiésemos hacer experimentos de seguridad). He de insistir en que estos mensajes de respuesta al **VERSION** pueden ser modificados por el usuario para que ponga cualquier cosa.

2.3.4.3. CTCP TIME

Por último, vamos a ver otro **CTCP** típico, que sirve para saber qué **hora** tiene el PC de un usuario. Este comando es útil para conocer el **país** en el que se encuentra el usuario (por la diferencia horaria), así como para sincronizar alguna cita. Aunque suene a coña, yo uso el TIME a veces para sincronizarme con algún amigo para quedar a alguna hora.

El formato del mensaje es muy simple:
PRIVMSG PyC :□TIME□

La respuesta a este comando será un **NOTICE** con éste formato:
PyC!~Pic@215-205-19-7.uc.nombres.ttd.es NOTICE fulanin : □TIME Fri Mar 28 13:29:16 2003□

2.3.5. El protocolo DCC

¡También el **DCC** funciona mediante el comando **PRIVMSG**! ¿Os gustaría saber cómo se pueden **interceptar** envíos de archivos, o chats privados, por DCC? ¿Os gustaría saber cómo **spoofear** vuestra IP en un DCC? Esto, y más, es lo que explicaré en el próximo artículo. ;-)

2.4. Comandos de consulta al servidor

Vamos a ver ahora un grupo de comandos que se utilizan para pedir información variada al servidor.

2.4.1. Mensaje del día (comando MOTD)

Cuando entramos en un server, nos da un mensaje de bienvenida, también conocido como **MOTD** (**M**essage **O**f **T**he **D**ay), que nos da información útil sobre el servidor. Podemos pedir al servidor en cualquier momento su MOTD, o incluso el de cualquier otro servidor de la misma red. Algunos son curiosos, como por ejemplo:

MOTD efnet.vuurwerk.nl
A ver a qué os recuerda ese dibujito ascii. ;-)

2.4.2. Carga del servidor (comando LUSERS)

Al principio del artículo os di algunas cifras concretas sobre los usuarios que había conectados en EfNet en ese instante, así como sobre el número de canales, y el número de usuarios en un servidor concreto. Todas esas cifras podemos verlas ejecutando: **LUSERS**

2.4.3. Versión del software en el servidor (comando VERSION)

No sólo podemos ver el software que utilizan los clientes, si no también el software del servidor, así como otra información útil, mediante el comando:
VERSION
Por ejemplo, en el servidor irc.isdnet.fr, ésta será la respuesta:

2.8/hybrid-6.3.1(20020418_1). irc.isdnet.fr AGHiKMpYZ TS5owc - WALLCHOPS PREFIX=(ov)@+ CHANTYPES=#& MAXCHANNELS=20 MAXBANS=25 NICKLEN=9 TOPICLEN=120 KICKLEN=90 NETWORK=EFnet CHANMODES=b,k,l,imnpst KNOCK MODES=4 are supported by this server

De aquí extraemos información interesante sobre la configuración del servidor. Por ejemplo, nos dice que sólo soporta canales de tipo **#** (globales) y **&** (locales) (existen también los canales **!** y **+**, pero no he considerado que merezca la pena hablar de ellos), el número máximo de canales por usuario es 20, el número máximo de entradas en la lista de bans (denegación de acceso a un canal) es de 25, la longitud máxima de un nickname es de 9 caracteres, la longitud máxima de un topic es de 120 caracteres, etc.

2.4.4. Estadísticas del servidor (comando STATS)

Muchos servidores desactivan alguna de estas opciones, al no querer que los usuarios puedan extraer demasiada información que podría ser utilizada para algún tipo de ataque. Son varias las opciones, pero mostraré sólo las que funcionan en el servidor que estamos utilizando de ejemplo.
En primer lugar, si hacemos:
STATS u
Nos muestra el **uptime** del servidor, es decir, cuánto tiempo lleva funcionando desde la última vez que se reinició.
En segundo lugar, está la estadística de **uso de comandos**, pero antes de que lo probéis os tengo que advertir. Si ejecutais este comando en este servidor os hará un **k-line** temporal, es decir, os expulsará de toda la red durante 24 horas. Eso sí, os lo hará después de daros el resultado, lo cual veo bastante absurdo. Si no quieren que utilices ese comando, ¿no es más razonable que no lo implementen, en lugar de ejecutarlo para después expulsarte? Para evitaros el k-line os pasteo aquí el resultado que me dio a mi la última vez que ejecuté este comando. El comando a ejecutar sería:
STATS m
Y ésta la respuesta del servidor:

ADMIN 161 1715
AWAY 1324744 41605335
CAPAB 8 160
CLOSE 6 20
CONNECT 491 9057
DIE 14 138
DLINE 5 119
ERROR 5425 156096
GLINE 1031 100586
HASH 4 8
HELP 124 442
HTM 54 7
INFO 356 4588
INVITE 624304 11705238
ISON 48403584 2260375326

JOIN 12335190 135226756
KICK 2169543 91710944
KILL 26745 2459300
KLINE 1210 73957
KNOCK 858 8501
LINKS 652 3195
LIST 75991 305704
LOCOPS 15 353
LTRACE 1 12
LUSERS 298179 7191563
LWALLOPS 0 0
MODE 72304763 1431760954
MOTD 467 3691
NAMES 127934 1381606
NICK 72403821 2147611586
NOTICE 36420416 2708147443
OPER 387 5729
OPERWALL 14733 536306
PART 11601855 106863938
PASS 1041568 7289166
PING 40404679 442423522
PONG 24647891 346314785
PRIVMSG 238041898 1376495144
QUIT 19063494 472024380
REHASH 15 89
RESTART 2 14
SERVER 4472 154251
SET 9 127
SJOIN 58209061 2226248553
SQUIT 4073 114713
STATS 819 7103
SVINFO 7 93
TESTLINE 6 94
TIME 86997 759067
TOPIC 1140668 80984008
TRACE 10218 59279
UNDLINE 0 0
UNGLINE 0 0
UNKLINE 26 757
USER 37767462 867701368
USERHOST 4102785 63264793
USERS 21238 327
VERSION 7628 55489
WALLOPS 176 8224
WHO 4503972 40190982
WHOIS 2268763 17289803
WHOWAS 6383 48645

Esos números nos muestran el número de veces que ha sido utilizado cada comando. La utilidad que podais sacar de esta información ya depende de vuestra imaginación. Por ejemplo, el uso más obvio es conocer todos los comandos que tiene implementados el servidor. Los comandos que no conozcais los podéis buscar en el **RFC 2812** y el **RFC 2813**, aunque algunos ni siquiera estarán contemplados en los RFC. Además, muchos de estos comandos sólo los pueden ejecutar los operadores de la red. Por si tenéis curiosidad, os pasteo a continuación el **k-line** que me hizo el servidor inmediatamente después de ejecutar el comando:

You are banned from this server- Temporary K-line 1440 min. - stop stating (2003/03/14 18.37)
-
[18:35] Closing Link: PyC[PyC@213.206.14.9] (Connection closed)

2.4.5. Comprobando la red (comando LINKS)

Con el comando **LINKS** podemos ver todos los servidores a los que está conectado el nuestro (o cualquier otro de la red), lo cual nos sirve para detectar posibles **splits** (desconexión de algunos servidores a la red, lo que causa la aparición de dos o más redes fantasma). Simplemente ejecutamos:
LINKS
Y nos dará la lista de servidores. Mediante este comando podemos estudiar además la topología de la red.

2.4.6. Poniendo el reloj en hora (comando TIME)

¿Alguna vez habéis utilizado el teletexto, o la radio para saber la hora y aseguraros de que

vuestro reloj no se ha quedado sin pilas? Pues se acabó el utilizar medios anticuados, a partir de ahora vais a poner en hora los relojes con vuestro servidor de IRC :P

TIME

Os dará la fecha y hora que tiene el servidor.

2.4.7. Traceo de servidores (comando TRACE)

Con este comando podéis ver la ruta que hay entre vuestro servidor y cualquier otro de la misma red. Si, por ejemplo, ejecutamos:

TRACE efnet.demo.co.uk

Nos mostrará la ruta que hay que atravesar para comunicar tu servidor con este otro por el camino más corto.
Nos puede servir también, por ejemplo, para buscar servidores de un determinado país. Por ejemplo, si hacemos:

TRACE *.es

Podremos comprobar que no hay ningún servidor español de EfNet. :-(

2.4.8. Información sobre los administradores (comando ADMIN)

Si queremos, por ejemplo, saber cómo contactar con los administradores del servidor irc.homelien.no, sólo tenemos que ejecutar:
ADMIN irc.homelien.no
Y nos dará la información de contacto.

2.4.9. Un poco de historia (comando INFO)

Si queremos saber algo de la historia de IRC, o del servidor, podemos ejecutar:

INFO
Para que nos cuenten lo que les de la gana. :-P

2.5. Comandos de consulta sobre los usuarios

Este último grupo de comandos son de uso cotidiano, no como los últimos que hemos visto. Son fundamentales, así que estad bien atentos. ;-)

2.5.1. Comando WHO

Este comando tiene muchas utilidades. Vamos a ver las más importantes, pero hay que tener en cuenta que los usuarios en modo invisible (modo de usuario +i) no aparecen en los listados generados por una consulta WHO.

2.5.1.1. Listado de usuarios

En primer lugar, podemos utilizar WHO para listar todos los usuarios de un determinado dominio:
WHO *.es
Nos mostrará todos los usuarios conectados a la red que vengan de un dominio .es, siempre y cuando no estén en modo invisible.

2.5.1.2. Listado de usuarios en canales

Para ver los usuarios (no invisibles) que hay en el canal #hackxcrack, haremos:
WHO #hackxcrack

2.5.1.3. Búsqueda de usuarios por IP

Este punto si que es realmente interesante. Si conocemos la IP de alguien, podemos saber si está conectado a nuestra red de IRC, y también con qué nickname.
Supongamos, por ejemplo, que la IP es:

217.125.24.17 (al igual que todas las IPs utilizadas a lo largo del artículo, esta IP ha sido escogida de forma totalmente aleatoria). En primer lugar, tenemos que conocer el **DNS** asignado a esa IP, para lo cual, desde nuestra aplicación cliente de IRC (¡no desde telnet!) podemos ejecutar lo siguiente:
/dns 217.125.24.17
Nos responderá algo como esto:
[17:04] * Looking up 217.125.24.17 -**
[17:04] * Resolved 217.125.24.17 to 217-125-24-17.uc.nombres.ttd.es**
Ya sabemos cual es el DNS asociado a esa IP. Si ahora hacemos la siguiente consulta:
WHO 217-125-24-17.uc.nombres.ttd.es*
Nos devolverá el (o los) nickname de esa IP, si es que ese usuario está conectado a la misma red de IRC que nosotros. Si el usuario no está conectado, simplemente nos dirá algo como esto:
217-125-24-17.uc.nombres.ttd.es* End of /WHO list.

2.5.2. Comando WHOIS

Este comando es uno de los que más utilizo constántemente. Nos da mucha información útil sobre un usuario. Si hacemos:
WHOIS PyC
Nos puede devolver una respuesta como ésta:
PyC is ~Pic2@138.Red-25-12-191.pooles.rima-tde.net * PyC PoC
PyC on #LCo @#spanishwarez @#hackxcrack +#españa
PyC using efnet.demon.co.uk <pils> how do you think i got oline?
PyC End of /WHOIS list.
La primera línea del whois nos indica su **nombre de usuario** (~Pic2), su **IP** o **DNS** (138.Red-25-12-191.pooles.rima-tde.net), y su **nombre "real"** (PyC PoC).
Después de eso, nos muestra la **lista de canales** en los que está el usuario, siempre y cuando el canal no sea **secreto** (modo +s) o **privado** (modo +p). Además, nos dice el tipo de usuario que es en cada canal. En este ejemplo concreto, el usuario PyC está como **usuario llano** en el canal #LCo, como **usuario con voz** en el canal #españa, y como **operador** en los canales #spanishwarez y #hackxcrack.
Por último, nos dice el **servidor** a través del cual está conectado el usuario. En este caso el servidor sería efnet.demon.co.uk. La frase que pone despues es un pequeño quote que añade el servidor, así que no tiene el más mínimo interés.
Si el usuario estuviese en el mismo servidor que nosotros, nos habría dado una información adicional, como ésta:
PyC has been idle 2secs, signed on Fri Mar 28 16:56:59
Esta información nos dice el **idle**, es decir, el tiempo que lleva este usuario sin escribir nada, y la fecha y hora en la que conectó con el servidor.
Si queremos saber esto mismo y no estamos en el mismo servidor que este usuario, bastará con que hagamos la consulta poniendo su nick 2 veces:
WHOIS PyC PyC
Intuitivo, eh? ;-)

2.5.3. Comando WHOWAS

Este comando nos dice cuándo estuvo conectado un determinado nick. Por ejemplo, para saber cuándo estuvo conectado el usuario PyC:
WHOWAS PyC
Nos responderá algo como esto:
PyC was PyC@203.47.23.8 * PoC
PyC using efnet.demon.co.uk Fri Mar 28 17:33:03 2003
End of WHOWAS

3. PARA TERMINAR

Buff... ha sido una tarea más pesada de lo que esperaba al principio, pero ya están resumidas las bases del protocolo IRC. :-)

En este artículo he seguido el RFC de forma más estricta que en los anteriores. Si habéis seguido los 3 artículos publicados hasta ahora, habréis comprobado que el estilo de redacción ha cambiado en los 3, y es debido a que estoy experimentando para ver vuestra reacción ante las distintas formas de escribir, para ver cual os gusta más. :-)

En primer lugar, he de aclarar que no he incluido la mayoría de las respuestas del servidor a cada comando, ya que si no el artículo se extendería hasta el infinito. He considerado que la mayoría de las respuestas son suficientemente intuitivas. Si lo queréis ver desde otro punto de vista, os dejo la interpretación de las respuestas como ejercicio. :-P

Se me ocurren algunas curiosidades que contar para terminar.

En primer lugar, os resumo la **lista de tareas que puede realizar el operador de un canal** (los de la famosa @):

- **INVITE** – invitar a un usuario a un canal en modo invite only (modo +i).
- **KICK** – expulsar a un usuario del canal.
- **MODE** – cambiar los modos del canal, y los modos de otros usuarios en el canal.
- **TOPIC** – cambiar el topic del canal si el canal está en modo +t. Si no, cualquiera puede hacerlo, y no solo un operador.
- **PRIVMSG** – Por supuesto, un operador siempre puede hablar en el canal, aunque éste se encuentre en modo moderado (modo +m).

Otra cosa que he de mencionar, es que no he hablado de la mitad del protocolo IRC, que es el que se utiliza entre los distintos servidores de una red, y no entre un cliente y un servidor. He dado por hecho que, si sois administradores de una red de IRC, tendréis mejores fuentes de consulta que esta revista y, en caso contrario, no os interesará lo más mínimo conocer este protocolo. De todas formas, lo tenéis completamente detallado en el **RFC 2813**.

Por último, una pequeña curiosidad, y es que el IRC no es case-sensitive, es decir, que no distingue entre mayúsculas y minúsculas, y esto produce algunas situaciones curiosas. Por ejemplo, por raro que parezca, en IRC se considera que los caracteres **{ } | ^** son las minúsculas de los caracteres **[ ] \ ~** respectivamente. Así que, por ejemplo, el nick ^{Zer0C00|}^ es equivalente al nick ~[zer0c00\]~. Si no sois asiduos a las redes de IRC, os parecerán nicks muy raros, pero os aseguro que mucha gente utiliza nicks con ese tipo de símbolos. :-)

Autor: PyC (LCo)

Agradecimientos: Scherzo (LCo), PoLLo, polhux, skitter, Tuxed, NetLander, _Stealth_, KaR]V[aN, y los que se me olviden.



MUY IMPORTANTE: COMUNICADO EDITORIAL

Este número 9 de PC PASO A PASO (Los Cuadernos de Hack x Crack) contiene gran cantidad de palabras y conceptos que seguro ha pillado desprevenido a más de uno. Si eres un lector habitual ya sabes que solemos explicar precisamente todos esos conceptos a lo largo de nuestros artículos, pero en este caso (y nos referimos en especial al artículo del NMAP) hemos preferido dar una visión general a partir de la cual profundizaremos en próximos números.

Si es la primera vez que nos lees y no posees un cierto nivel en esto de la informática, seguro que te has quedado "fuera de lugar" y sin saber muy bien qué tipo de revista has comprado, por ejemplo, hablamos de establecer conexiones por TELNET sin más explicaciones y nos quedamos "tan anchos". Bien, esto no es así, en los números anteriores hemos explicado hasta el hartazgo desde cómo se abre una Ventana DOS (Ventana de Comandos) hasta cómo se establece una conexión por Telnet y mil cosas más de nivel básico. Estás leyendo un número que utiliza los conocimientos ya adquiridos en anteriores entregas, por ejemplo el curso de Visual Basic ya está por la entrega 3, igual que la SERIE RAW.

Si quieres ver nuestra línea editorial tienes en nuestra Web el número 1 de Hack x Crack a tu disposición en formato PDF, descárgalo y léelo tranquilamente, es completamente gratis y podrás ver que hemos empezado DESDE CERO!!!

Posdata para los miembros del foro: Si algún miembro del foro cree que esta nota la hemos puesto para "rellenar", solo tiene que mirar la revista para darse cuenta de que hemos empezado a reducir el tamaño de letra porque hemos tenido verdaderos problemas de espacio, no digamos ya los concursos de Linux y logos (que son miniaturas en comparación con anteriores números). Esta nota es muy importante porque tenemos miedo de estar tomando una línea demasiado técnica y todos sabemos que eso es peligroso, por ello necesitábamos advertir a los nuevos lectores que si no comprenden este número de PC PASO A PASO es precisamente porque son nuevos lectores y no han seguido la revista desde su inicio.

Parece mentira pero en tan solo 9 números nos hemos introducido de lleno en un mundo para muchos desconocido e intentamos en cada nuevo número editado llegar un poco más lejos. Ahora sería el momento ideal para decirte que si te falta algún número de la revista puedes pedirlo desde nuestra Web e intentar venderte "la moto", pero no es ninguna "moto", es algo que deberás hacer si eres un nuevo lector y quieres seguir avanzando. Para nosotros la venta de números atrasados es más una cruz que un beneficio, te lo aseguro, pero es un servicio que estamos dando porque es IMPRESCINDIBLE para los nuevos lectores. Bueno, que me quedo sin espacio, un fuerte abrazo a toda la peña del foro ;)